Univerza v Ljubljani
Fakulteta za računalništvo in informatiko

Uroš Stegel

Sistem za merjenje časa na atletskih stezah

diplomsko delo
na univerzitetnem študiju

Mentor: prof. dr. Nikolaj Zimic

Ljubljana, 2016

Besedilo je oblikovano z urejevalnikom besedil LaTeX.

Fakulteta za računalništvo in informatiko izdaja naslednjo nalogo:

Tematika naloge: Sistem za merjenje časa na atletskih stezah

Elektronsko merjenje časa na atletskih stezah predstavlja zanimiv izziv, saj je natančnost meritve bistvenega pomena. Na čas vpliva veliko dejavnikov, kot na primer zajem dogodka in nenazadnje tudi vpliv vremenskih dejavnikov ter temperature.

V diplomski nalogi izdelajte sistem za merjenje časa na atletskih stezah. Sistem naj sestavljajo nadzorna enota ter enote, ki so postavljene na posameznih merilnih točkah. Vsi elementi sistema naj bodo med seboj povezani z brezžično povezavo. Dostop do nadzorne enote naj bo omogočen preko mobilnega telefona.

Ker je sistem časovno kritičen, opredelite vse časovno kritične faze, ki lahko vplivajo na končno meritev. Za tem izdelajte meritev, s katero boste potrdili pravilnost delovanja sistema v predvidenih časovnih okvirih.

Izjava o avtorstvu diplomskega dela

Spodaj podpisani Uroš Stegel sem avtor diplomskega dela z naslovom:

Sistem za merjenje časa na atletskih stezah (angl. *Track and Field time measurement system*)

S svojim podpisom zagotavljam, da:

- sem diplomsko delo izdelal samostojno pod mentorstvom prof. dr. Nikolaja Zimica,
- so elektronska oblika diplomskega dela, naslov (slov., angl.), povzetek (slov., angl.) ter ključne besede (slov., angl.) identični s tiskano obliko diplomskega dela,
- soglašam z javno objavo elektronske oblike diplomskega dela na svetovnem spletu preko univerzitetnega spletnega arhiva.

V Ljubljani, dne 15. junija 2016 Podpis avtorja:

Zahvaljujem se mentorju prof. dr. Nikolaju Zimicu za strokovno pomoč, usmerjanje in svetovanje pri izdelavi diplomske naloge. Zahvaljujem se tudi Petru Koroščcu za doprinos pri načrtovanju in razvoju rešitve, Zorotu Nedeljkoviču za pomoč pri meritvah in svoji družini, ki je verjela vame in me spodbujala skozi vsa ta leta.

Mojemu Pačiju, Nini, Črtu, Boru.

Kazalo

Seznam uporabljenih kratic

kratica	angleško	slovensko
EEPROM	electrically erasable programmable read-only memory	električno izbrisljiv programirljiv bralni pomnilnik
LCD	liquid-crystal display	zaslon s tekočimi kristali
SPI	serial peripheral interface	zunanji serijski vmesnik
UART	universal asynchronous receiver/transmitter	univerzalni asinhroni sprejemnik in oddajnik
RISC	reduced instruction set computer	računalnik s skrčenim naborom ukazov
MIPS	million instructions per second	milijon ukazov na sekundo
ARM	advanced RISC machine	mikroprocesor RISC
ISM	industrial, scientific and medical radio bands	industrijsko, znanstveno medicinski radijski pasovi
PLL	phase-locked loop	fazno zaključena zanka
FSK	frequency shift keying	modulacija s frekvenčnim skokom
OOK	on-off keying	modulacija s prižiganjem in ugašanjem nosilca

kratica	angleško	slovensko
LNA	low noise amplifier	nizkošumni ojačevalnik
RSSI	received signal strength indication	indikacija moči prejetega signala
GFSK	Gaussian FSK	Gaussov FSK
MSK	minimum shift keying	modulacija z minimalnim skokom
GFSK	Gaussian MSK	Gaussov MSK
FIFO	first-in-first-out	prvi-noter-prvi-ven
CRC	cyclic redundancy check	ciklična redundančna koda
NRZ	nonreturn to zero	brez vračanja na 0
TSR	transmit shift register	oddajni pomikalni register
RF	radio frequency	radijska frekvenca
TCXO	temperature compensated crystal oscillator	temperaturno kompenziran kvarčni oscilator
ASCII	american standard code for information interchange	ameriška standardna koda za izmenjavo informacij
IR	infrared	infrardeč
PCM	pulse code modulation	pulzno kodna modulacija
NOR	NOT-OR logical op	NE-ALI logični operator
MVC	Model–view–controller	model-pogled-kontroler
RFCOMM	Radio frequency communication	radijsko frekvenčna komunikacija

Povzetek

Naslov: Sistem za merjenje časa na atletskih stezah

Meritve časa pri športih, katerih osnova je hitro gibanje, zahtevajo visoko natančnost. Pri najpogosteje uporabljenem ročnem merjenju časa temu pogoju ni mogoče zadostiti. Napake zaradi človeškega faktorja so prevelike in se razlikujejo med osebami, ki meritve opravljajo.

Tem napakam se lahko izognemu z uporabo elektronskih senzorjev, ki dogodke vzorčijo z visoko natančnostjo.

V diplomski nalogi smo zasnovali porazdeljen sistem za elektronsko merjenje časa. Sestavljajo ga nadzorna enota ter enote s senzorji, ki so med seboj povezane z brezžično povezavo. Enote imajo lastne ure, ki jih je potrebno pred meritvami časovno uskladiti. Dogodke prekinitev v gibanju vzorčimo s senzorskimi vrati z infrardečim žarkom, medtem ko vzorčimo štartne dogodke s tračnim stikalom.

Pri razvoju smo dali pozornost kritičnim elementom, ki vplivajo na končno napako meritev: natančnosti ur, časovnemu usklajevanju enot in zajemanju dogodkov s senzorji.

Na koncu smo s testi potrdili, da omenjen sistem zadostuje zahtevanemu pogoju za natančnost, in sicer, da je le-ta vsaj 10-krat višja od meritvene resolucije, ki znaša 0,01 s.

Ključne besede: meritev, čas, infrardeč, brezžičen, sistem, natančnost, senzor, šport.

Abstract

Title: Track and Field time measurement system

Time measurement in sports, which are based on speed and fast movement, require high accuracy. In the most commonly used manual timing this requirement cannot be met. Errors due to human factor are too high and vary between persons who carry out measurements. These errors can be avoided by using electronic sensors, which sample events with high accuracy.

In this thesis, we designed a distributed system for electronical time measurement. It consists of the control unit and sensor units that mutually communicate over radio frequencies. Units have their own clocks, which must be synchronized before measurements.

Events triggered by moving objects are captured by infrared beam gates, while a special switch is responsible for sampling starting events.

In the development we have given attention to the following elements that affect the accuracy of measurement : clock accuracy, time synchronisation and sensor event capturing.

Finally, we experimentally confirmed that such a system suffices accuracy measures stating that accuracy is at least 10 times the measurement resolution being 0.01 s.

Keywords: measurement, time, infrared, wireless, system, accuracy, sensor, sport.

Poglavje 1

Uvod

Namen dela je razviti elektronski sistem za natančno merjenje časa v športu. Predvsem je tu mišljeno merjenje športnih disciplin, povezanih s hitrim gibanjem, kjer je zahtevana visoka natančnost meritve. To dosežemo z uporabo elektronskih senzorjev, ki generirajo prekinitvene dogodke z veliko manjšo napako, kot to omogočajo ročne merilne naprave. Vrata z infrardečim žarkom predstavljajo senzor, ki generira prekinitvene dogodke v trenutku, ko merjen objekt oz. subjekt v gibanju prekine IR-žarek, medtem ko uporabljamo posebno tračno stikalo za generiranje prekinitvenih dogodkov, povezanih z začetkom gibanja (štartom) subjekta.

Zantevana natančnost je vsaj 10-krat višja od resolucije merjenja, ki znaša 0,01 s. Sistem sestavlja centralna enota ter poljubno število senzorjev. Gre za procesno ter podatkovno porazdeljen sistem, kjer ima vsaka enota svojo lokalno uro. Enote med seboj komunicirajo preko radijskih valov.

Naloga senzorskih enot je vzorčenje prekinitvenih dogodkov, ki jih prožijo njihovi senzorji. Dogodke je potrebno opremiti s trenutno vrednostjo relativne ure. Za to ima senzorska enota na voljo lokalno uro, ki pa mora biti predhodno časovno usklajena z uro centralne enote. Relativna ura predstavlja čas, ki je potekel od trenutka zagona centralne enote in je orientacija za vse nadaljne meritve.

Preko centralne enote upravljamo z meritvenim sistemom. Enota omogoča

funkcionalnosti, kot so upravljanje s kronometrom, pregledovanje hranjenih informacij o prekinitvenih dogodkih ter kontrolo nad povezovanjem s senzorskimi enotami. Enota v procesu merjenja zbira podatke o prekinitvenih dogodkih s strani senzorskih enot ter jih podaja v obdelavo ter shranjevanje funkciji kronometra. Meritveni dogodki se v procesu zajemanja ne smejo izgubiti.

V središču našega zanimanja je zagotavljanje zahtevane natančnosti meritev. Izbrati je potrebno urine izvore z dovolj veliko natančnostjo. Ti morajo biti v čim večji meri odporni na temperaturne spremembe. Uporabiti je potrebno časovno deterministične protokole za komunikacijo, kajti ure senzorskih enot je potrebno časovno usklajevati z uro centralno enoto. Na koncu so tu še napake samih senzorjev, ki generirajo prekintvene dogodke.

Potem so tu problemi okoli komunikacije, kot so zadosten radijski doseg ter sposobnost komunikacije ob prisotnosti več sorodnih kakor tudi drugih sistemov na lokaciji merjenja.

Motiv za razvoj meritvenega sistema izhaja iz sveta športa. Predvsem gre tu za športe, kjer je osnova gibanje v hitrosti, kot so to atletika, nogomet, košarka itd. V procesu treninga so večkrat potrebna razna merjenja hitrosti, s katerimi preverjamo trenutno kondicijsko pripravo. Problem se pojavi glede dostopnosti do natančnih meritvenih sistemov. Podobni sistemi na tržišču sicer že dolgo obstajajo, vendar so zaradi visokih cen posamezniku skoraj nedostopni. V večji meri so last raznih organizacij kot so klubi oz. društva.

Cilj je, da bi se tako rešitev čim bolj približalo posamezniku, seveda s čim nižjimi razvojnimi stroški. Tu imamo v mislih predvsem povezovanje sistema preko pametnih telefonov, saj v tem primeru del stroškov okoli uporabniškega vmesnika odpade.

Diplomsko delo obsega pet poglavij. V prvem poglavju beremo uvodno besedilo. V drugem poglavju se dodaknemo tematike v teoretičnem pogledu. Govorimo o vrstah testiranj, ki jih lahko opravljamo s sistemom ter zakaj so omenjena testiranja tako zelo pomembna. Nekaj besed namenimo ročnim meritvam ter njihovim pomanjkljivostim. Sledi opis naprave in posebnosti

pri merjenju. Največji povdarek pa gre natančnosti meritvenega sistema.

Tretje poglavje opisuje strojno opremo sistema. Predstavljeni sta matični plošči centralne in senzorske enote. Sledijo opisi osnovnih značilnosti posameznih komponent, ki sestavljajo te plošče. Omenjen je njihov morebitni doprinos k napaki meritev. Sem spadajo mikrokontroler serije PIC18, radijski modul rfm69w, bluetooth modul rn42 itd.

Četrto poglavje govori o programski opremi. Sem spada opis operacijskega sistema z osrednjim modulom, ki skrbi za prenašanje sporočil in izvajanje akcij nad objekti glede na prioriteto procesiranja. Sledi opis komunikacijskega sklada, potem opisi gonilnikov, ki skrbijo za delovanje v tretjem poglavju omenjenih komponent. Nazadnje imamo opis same aplikacije za izvajanje časovnih meritev ter povezljivosti sistema z zunanjimi napravami.

V zadnjem, petem poglavju, ki je namenjeno testiranju preizkusimo izdelan prototip meritvenega sistema v realnem okolju. Pri tem opravimo meritve, kot so izmera velikosti radijskega dosega, odstopanje pri časovnih uskladitvah ter absolutna napaka ur na določenih časovnih intervalih v različnih temperaturnih pogojih. Nadvsezadnje sledi primerjava merjenja z referenčno napravo proizvajalca Brower, ki je na voljo na tržišču.

Poglavje 2

Merjenje časov v atletiki

2.1 Preverjanje pripravljenosti atletov

Cilj diplomske naloge je izdelati sistem za merjenje časa v atletiki. Predstavljena je rešitev, ki bo v pomoč atletom pri preverjanju hitrosti. Gre predvsem za merjenja pri izvajanju raznih vaj, povezanih s tekom. Najpogosteje se merjenja opravljajo proti koncu pripravljalnega obdobja, tik preden se začnejo tekmovanja. Na podlagi rezultatov le-teh si trenerji pomagajo pri planiranju nadaljnih treningov in izbiri tekmovanj.

Sposobnost doseganja visokih hitrosti pri teku je pomembna lastnost, ki jo morajo osvojiti ne le tekači na kratke proge, temveč tudi atleti drugih disciplin. Odvisna je od fizične kondicije in pa tehnične izvedbe teka. Trenerji vsakega atleta obravnavajo na svojstven način. Pri tem spremljajo več različnih parametrov, kot so:

- relativna moč atleta (glede na telesno težo),
- elastičnost v skočnem zglobu,
- kontaktni čas stopala na podlago,
- odrivna moč, katere pokazatelji so skoki z noge na nogo, skoki po eni nogi, skoki z mesta v daljino, višino itn.,

- raztegljivost posameznih mišičnih skupin, kar pripomore k dolžini koraka,
- eksplozivnost z malimi in velikimi bremeni, pomembna je predvsem pri pospeševanju,
- frekvenca vklapljanja posameznih mišičnih skupin ...

Tako obstajajo različne vrste testov, pri katerih lahko opravljamo meritve časa. Merimo lahko maksimalno hitrost pri normalnem teku, pri teku na dolg korak oz. teku na frekvenco. Glede na konstrukcijo ter trenutno fizično kondicijo atleta se pri teku v maksimalni hitrosti vzpostavi optimalno razmerje med dolžino koraka in njegovo frekvenco. Produkt omenjene dolžine koraka ter frekvence je maksimalna hitrost. Torej tek na frekvenco v atletskem žargonu pomeni, da poskuša atlet kar se da najhitreje teči, pri čemer je frekvenca teka višja od optimalne frekvence pri maksimalni hitrosti. Obratno velja pri teku na korak, ko poskuša atlet pri teku doseči čim višjo hitrost, pri čemer je dolžina koraka večja od optimalne dolžine pri maksimalni hitrosti. Potem lahko merimo hitrosti pri teku z bremeni, bodisi z obtežitvijo atleta samega bodisi z vlečenjem sani oz. potiskanjem vozičkov. Merimo lahko hitrosti poskokov z noge na nogo oz. po eni nogi.

Najpogosteje se meri t. i. leteča hitrost med točkama A in B, ki sta med seboj oddaljeni 30 m. Šprinter naj bi maksimalno hitrost ob polnem šprintu razvil med 40. in 70. metrom (razvidno iz tabele 2.1). Po 70. metru začne hitrost postopoma upadati. Hitrost upadanja je odvisna od komponente "vzdržljivosti pri maksimalni hitrosti" (tabela 2.1 — označeno z rumeno). To lahko izmerimo na več različnih načinov. Lahko merimo letečo hitrost med točkama A in B, oddaljenima od 80 do 150 metrov. Leteča hitrost v atletskem žargonu pomeni maksimalno povprečno hitrost, ki jo je sposoben atlet držati med dvema določenima točkama, pri čemer vstopa v interval s tako hitrostjo, da je rezultat optimalen (slika 2.4). Uporabimo lahko več točk, med katerimi se meri hitrost. Npr. po 30-metrskem intervalu (A,B) postavimo več točk, med seboj oddaljenih 10 m, tako da je zadnja točka od

	Ben '88	Carl '88	Mo '99	Mo '01	Tim '02	Asafa '05	BOLT '08
RT	0.132	0.136	0.162	0.132	0.104	0.150	0.165
0-10m	1.83	1.89	1.86	1.83	1.89	1.89	1.85
10-20m	1.04	1.07	1.03	1.00	1.03	1.02	1.02
20-30m	0.93	0.94	0.92	0.92	0.91	0.92	0.91
30-40m	0.86	0.89	0.88	0.89	0.87	0.86	0.87
40-50m	0.84	0.86	0.88	0.86	0.84	0.85	0.85
50-60m	0.83	0.83	0.83	0.83	0.83	0.85	0.82
60-70m	0.84	0.85	0.83	0.83	0.84	0.84	0.82
70-80m	0.85	0.85	0.86	0.86	0.84	0.84	0.82
80-90m	0.87	0.86	0.85	0.89	0.85	0.85	0.83
90-100m	0.90	0.88	0.85	0.91	0.88	0.85	0.90
TIME	9.79	9.92	9.79	9.82	9.78	9.77	9.69
			Courtesy of SpeedEndurance.com				

Slika 2.1: Primerjava časov meritev znanih atletov pri teku na 100 m, merjeno na intervalih po 10 m. Z rumeno barvo so prikazani časi Usaina Bolta med 50. in 90. metrom, povezani z vzdržljivostjo pri maksimalni hitrosti.[2]

točke A oddaljena 150 m.

Eden izmed pristopov k merjenju pri treningih izboljševanja frekvence šprinterskega koraka je ta, da izmerimo hitrost na intervalu (A, B), pogojeno z določenim številom napravljenih korakov. Dolžina koraka mora biti krajša kot pri običajnem šprintu.

Pri izboljšavah štartnega pospeška merimo čas pri štartih z mesta. Lahko jih opravimo bodisi z raznimi bremeni (vlečenje obteženih sani nad 5—20 kg) oziroma brez.

2.2 Merjenje z ročnimi časomerilnimi napravami

Meritve z ročnimi merilnimi napravami niso tako zanesljive zaradi prevelikih napak. Še vedno jih prakticirajo na uradnih tekmovanjih kot podporne

[2]Vir:Usain Bolt 100 m 10-metrski preseki in hitrostna vzdržljivost (vzeto s strani dne 11. 3. 2016) `http://speedendurance.com/2008/08/22/usain-bolt-100m-10-meter-splits-and-speed-endurance/`

meritve v primeru izpada elektronske metode oz. v primeru, ko elektronski sistem ni na voljo. Uradno tem rezultatom pribijejo 2.4 s kot povprečno napako meritve. Ne nastopajo pa ti rezultati kot kandidati pri beleženju rekordnih dosežkov.

Največje napake pri ročnem merjenju napravimo v točkah, kjer se atlet giblje najhitreje. Če merimo čase na odsekih, je lahko takih točk več in posledično je napaka meritve večja. Meritve z elektronskin sistemom te pomanjklivosti nimajo.

Kot druga težava pri ročnem merjenju se izpostavi zahtevana prisotnost trenerja oz. druge osebe, ki meritev opravlja, kar pa ni vedno izvedljivo. Atlet lahko elektronski sistem postavi in uporablja sam.

Oviro pri ročnem merjenju predstavljajo tudi razdalje. Še posebej, če je atletov, ki jih merimo na različnih razdaljah več. Največkrat so ciljne točke meritev različne in je za merilca praktično nemogoče opazovati prehode atletov skozi ciljne točke iz prave perspektive. Atlet ponavadi vzdigne roko visoko v zrak, v znak merilcu, da je prečkal ciljno črto.

Pri začetku merjenja teka se lahko merilec osredotoča bodisi na prvi gib atleta oz. trenutek, ko atlet z nogo napravi prvi korak. Slednja metoda nam omogoča natančnejša merjenja, ker lahko trenutek dotika tal z nogo pričakujemo glede na prvi gib atleta, medtem ko nas pri prvi metodi sam štart lahko preseneti. Kljub boljši natančnosti je zaradi drugačne izvedbe primerjava oz. napoved rezultata na tekmah manj zanesljiva.

2.3 Naprava za merjenje časa

Sistem za merjenje sestoji iz senzorjev in centralne enote, ki jih povezuje. Centralna enota, kakor tudi senzorji imajo svojo uro. Ure je potrebno med seboj časovno uskladiti. Čas usklajenih ur je relativen. Gre za čas, ki je potekel od zagona centralne enote (uptime).

Senzorji beležijo dogodke pri meritvah. Ti dogodki predstavljajo začetni, vmesni ali končni čas meritev. Centralna enota skrbi za zajem in obdelavo

podatkov o dogodkih. Obdelani podatki so nato na voljo uporabniku preko uporabniškega vmesnika.

Centralna enota in senzorji med seboj komunicirajo preko radijskih valov. Komunikacijski protokol mora zagotoviti, da se informacije o beleženih dogodkih ne izgubijo.
Vrste senzorjev:

- vrata z infrardečim snopom,
- stikalo v obliki traku,
- zvočno stikalo za uporabo štartne pištole.

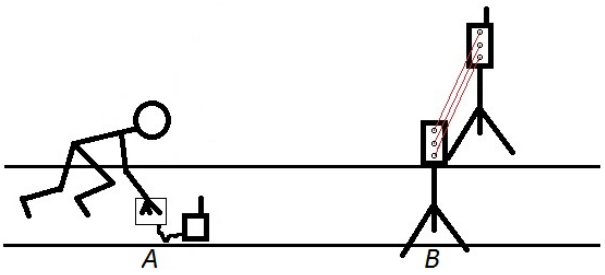
Merilna vrata sestavljata oddajnik in sprejemnik usmerjenega žarka infrardeče svetlobe. IR-žarek potuje od oddajne diode do sprejemnika. V kolikor je žarek prekinjen, smatramo to za dogodek meritve.

Slika 2.2: Določitev meritvenih dogodkov z vertikalnimi časovnicami za atlete v fotofinišu. [4]

Z vrati z IR-snopom lahko naenkrat merimo samo enega atleta. Razlog je ta, da lahko atleti pri prehodu skozi točko merjenja drug drugega zakrivajo. To bi bilo mogoče, če bi namesto IR-snopa uporabili kamero z visoko

[4]Fotofiniš ženskega teka na 100 m. (vzeto s strani dne 10. 3. 2016) `http://styleofsport.com/it-was-a-photo-finish/`

frekvenco vzorčenja. Vsako sliko bi nato uparili s trenutnim časom. Naknadno bi moral uporabnik, prav tako kot to počno sodniki na tekmah, določiti dogodke meritve za vsakega od merjencev glede na časovne vertikale, kot je razvidno na sliki 2.2. To pa presega obseg diplomske naloge.



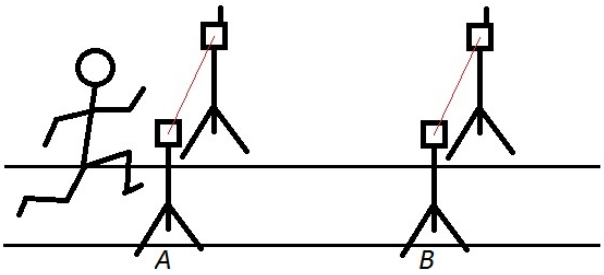
Slika 2.3: Merjenje časa atleta med točko A, ki predstalja štart in točko B. Prikazana uporaba tračnega stikala in pa IR-celic s tremi žarki.

Omeniti velja težavo, do katere pride zaradi predčasne prekinitve žarka z roko merjenca. Dogodek, ki oznanja prehod skozi vrata, naj bi se zgodil v času, ko merjenec s trupom prečka točko merjenja. Velikost napake ocenimo nekje na intervalu (0, 50) v centimetrih. Absolutna časovna napaka je odvisna od hitrosti atleta v času napake. Pristop, ki ga lahko uberemo tukaj, je razporeditev več žarkov po vertikalni črti skozi črto merjenja, kot je prikazano na sliki 2.3. Dogodek se v tem primeru zavede ob prekinitvi vseh žarkov.

Stikalo v obliki traku služi kot senzor štartnega dogodka. Atlet v t. i. nizkem štartu položi palec ene izmed rok na trak. Preden atlet štarta, se mora tračni senzor aktivirati. To pomeni, da mora atlet počakati 2 s ob tem, da drži palec na traku (stikalo sklenjeno). Dogodek beležimo v trenutku štarta, ko atlet z roko zapusti tla (stikalo se razklene).
Sistem omogoča merjenje:

- časa teka v maksimalni hitrosti na razdalji med točkama A in B (kot je to razvidno na sliki 2.4); iz meritve lahko izračunamo maksimalno povprečno hitrost atleta.

- časa, ki ga atlet potrebuje iz točke A, kjer je v mirovanju, da kar

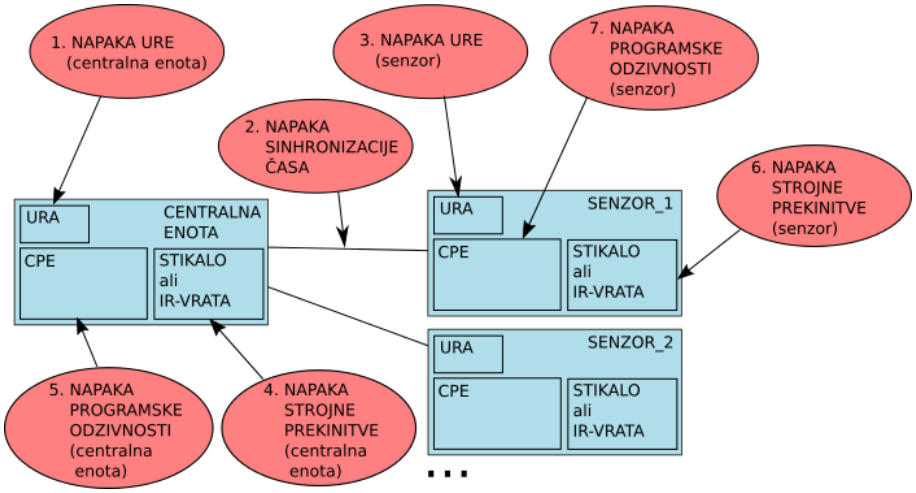


Slika 2.4: Meritev maksimalne hitrosti atleta na odseku med točkama A in B z merilnimi vrati z infrardečim snopom.

najhitreje preteče do točke B; tu lahko izračunamo povprečni štartni pospešek (slika 2.3).

- in pa časa reakcije; to je čas med dogodkom, ki predstavlja štartni signal in pa dogodkom, do katerega pride, ko atlet zapusti štartno mesto.

2.3.1 Natančnost



Slika 2.5: Sedem različnih napak, iz katerih sestoji skupna napaka pri meritvi.

Rezultati meritev v atletiki se beležijo z resolucijo 1/100 s. Enako velja za šprinterske discipline kakor tudi v maratonu. V primeru nejasnosti v vrstnem redu si pri razvrščanju pomagajo s fotofinišem. Resolucija pri fotofinišu

je 1/1000 s. V motošportih zaradi velikih hitrosti največkrat uporabljajo resolucijo meritve 1/1000. V smučanju, kljub temu da so hitrosti velike, je resolucija meritve 1/100.

Lokalna ura predstavlja število časovnih enot, ki je preteklo od trenutka zagona centralne enote. Centralna enota in senzorji imajo svoje ure. Te ure imajo relativno napako $relN[ura_{enota_x}]$[5].

Dogodke prekinitev vzorčimo na centralni enoti in senzorjih. Vsak dogodek opremimo z informacijo o vrednosti ure na dotični enoti v trenutku prekinitve T_{pr}. Meritev sestoji iz dogodka, ki predstavlja začetek meritve, nič ali več dogodkov, ki predstavljajo vmesne prekinitve ter dogodka, ki meritev zaključuje.

Časovna usklajevanja opravljamo inkrementalno. Uskladitev se na centralni enoti zavede kot dogodek, uparjen z informacijo o vrednosti časa centralne ure v trenutku sinhronizacije T_{sync} ter informacijo o množici senzorjev, katerih ura je bila usklajena v dotični sinhronizaciji. Tako obstaja več podmnožic senzorjev, ki pripadajo različnim sinhronizacijskim dogodkom.

Zahtevana skupna napaka pri meritvi z našim sistemom mora biti manjša od 1/100 s na vsakem meritvenem intervalu. Meritvena napaka je odvisna od napake posameznih ur, napake pri sinhronizaciji časa, strojne napake fizične prekinitve ter napake programske odzivnosti, kot je prikazano na sliki 2.5.

Napaka meritve se ocenjuje glede na t. i. referenčni dogodek centralne enote. Ta predstavlja dogodek uparjen z najmanjšim časom, ki sodeluje v meritvi. Bodisi gre za prekinitveni dogodek, ki izvira na centralni enoti bodisi za dogodek časovne uskladitve, če vsaj en senzor izmed pripadajoče podmnožice nastopa kot izvor enega ali več prekinitvenih dogodkov meritve. Čas referenčnega dogodka označimo s T_{ref}.

Napako ure centralne enote od zagona do referenčnega dogodka zanemarimo. Privzamemo torej, da je vrednost ure referenčnega dogodka brez napake.

[5]indeks $_{enota_x}$ predstavlja eno izmed enot iz množice $\{ce = enota_0, s_0 = enota_1, \ldots, s_N = enota_{(N+1)}\}$, kjer enota z indeksom 0 predstavlja centralno enoto, enote z indeksom 1..(N+1) pa senzorje

Napaka prekinitvenega dogodka na centralni enoti $N_{pr_{ce}}$ sestoji iz absolutne napake ure centralne enote $N[ura_{ce}]_{pr}$, nastale v času $T_{pr_{ce}} - T_{ref}$, napake strojne prekinitve $N[hw_{ce}]_{pr}$ ter napake zaradi programske odzivnosti na prekinitev $N[sw_{ce}]_{pr}$.

Napaka prekinitvenega dogodka na senzorski enoti N_{pr_s} sestoji iz absolutne napake ure centralne enote $N[ura_{ce}]_{sync_s}$ v trenutku sinhronizacije, nastale v času $T_{sync_s} - T_{ref}$, napake pri sinhronizaciji $N[sync_s]$, absolutne napake ure senzorja, nastale od sinhronizacije do prekinitve $N[ura_s]_{pr}$, napake strojne prekinitve $N[hw_s]_{pr}$ ter napake zaradi programske odzivnosti na prekinitev $N[sw_s]_{pr}$.

$$\begin{aligned} N[ura_{ce}]_{pr} &= (T_{pr_{ce}} - T_{ref}) * relN[ura_{ce}] \\ N[ura_s]_{pr} &= (T_{pr_s} - T_{sync_s}) * relN[ura_s] \\ N_{pr_{ce}} &= N[ura_{ce}]_{pr} + N[hw_{ce}]_{pr} + N[sw_{ce}]_{pr} \\ N_{pr_s} &= N[ura_{ce}]_{sync} + N[sync_s] + N[ura_s]_{pr} + N[hw_s]_{pr} + N[sw_s]_{pr} \end{aligned} \tag{2.1}$$

Izmerjen čas med dvema prekinitvenima dogodkoma meritve je $T_{pr_{enota_i}} x - T_{pr_{enota_j}} y + N_{pr_{enota_i}} x - N_{pr_{enota_j}} y$. Napaka meritve med dvema dogodkoma je razlika med napakama dogodkov A in B. Če npr. dogodka A in B izvirata iz istega senzorja potem je napaka neodvisna od napak ure centralne enote in sinhronizacije časa, kot je razvidno iz izračuna 2.2.

$$
\begin{aligned}
N_{pr_s}x &= N[ura_{ce}]_{sync} + N[sync_s] + N[ura_s]_{pr}x + N[hw_s]_{pr}x + N[sw_s]_{pr}x \\
N_{pr_s}y &= N[ura_{ce}]_{sync} + N[sync_s] + N[ura_s]_{pr}y + N[hw_s]_{pr}y + N[sw_s]_{pr}y \\
N_{[x,y]} &= N_{pr_s}x - N_{pr_s}y \\
N_{[x,y]} &= N[ura_s]_{pr}x + N[hw_s]_{pr}x + N[sw_s]_{pr}x \\
&\quad -N[ura_s]_{pr}y + N[hw_s]_{pr}y + N[sw_s]_{pr}y
\end{aligned}
\tag{2.2}
$$

Natančnost urinega takta merimo v enotah ppm (kosov-na-milion, $N \cdot 10^{-6}$) oz. ppb (kosov-na-miliardo, $N \cdot 10^{-9}$). Pri natančnosti N ppm trdimo, da po preteku 10^6 kosov dejansko beležimo od (10^6 - N) do (10^6 + N) kosov. Kos v tem primeru lahko predstavlja katerokoli časovno enoto. Tako po preteku 11 dni, 13 ur, 46 minut in 40 sekund (kar predstavlja 1000000 sekund) naprava, ki jo poganja kristal z natančnostjo ± 2 *ppm*, beleži dejanski čas te vrednosti plus napako v velikosti ± 2 sekund.

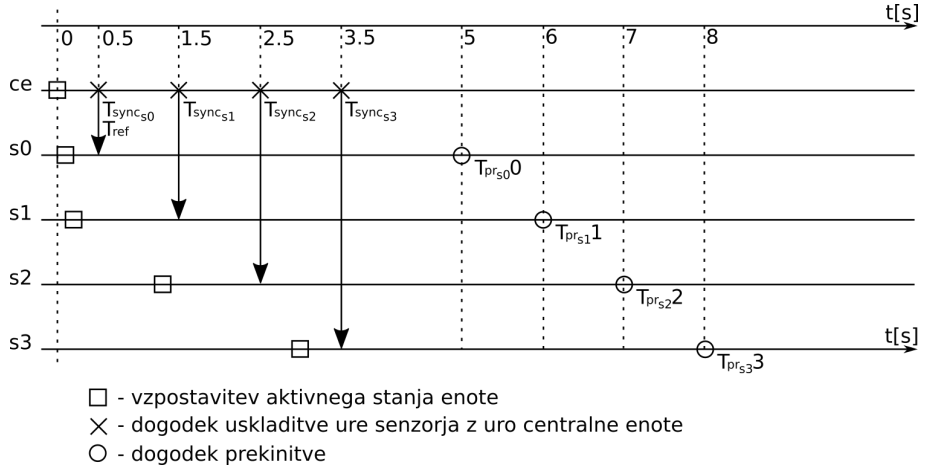
Večja kot je natančnost takta ure, višji so stroški enote. Stroške lahko optimiziramo s tem, da izberemo manj natančne frekvenčne oscilatorje ure senzorjev. S tem smo sicer primorani povečati frekvenco sinhronizacij.

Kot primer vzamemo meritveni sistem, katerega slika 2.6 prikazuje diagram dogodkov v času. Sestavljajo ga centralna enota (ce) ter štirje senzorji ($s_0..s_3$). Poskušamo predstaviti izračun maksimalne napake pri meritvi leteče hitrosti skozi štiri prekinitvene IR senzorje, ki so med seboj oddaljeni 10 m. Za oceno napak potrebujemo neodvisno relativno časovno komponento brez napake.

Privzamemo, da atlet teče s hitrostjo 10 m/s skozi celotno dolžino merjenja brez odstopanj pri hitrosti. Čas med poljubnima sosednima dogodkoma prekinitev je tako točno 1 s.

Privzamemo parametre, ki jih prikazuje naslednja tabela.

	$relN[ura_{enota_x}]$	$N[sync_{s_x}]$	$N[hw_{s_x}]_{pr}$	$N[sw_{s_x}]_{pr}$
ce	$\pm 2ppm$			
s_0	$\pm 2ppm$	$\pm[5/32768]s$	$\pm[1/6] \cdot 10^{-3}s$	$\pm 2 \cdot 10^{-6}s$
s_1	$\pm 7.5ppm$	$\pm[5/32768]s$	$\pm[1/6] \cdot 10^{-3}s$	$\pm 2 \cdot 10^{-6}s$
s_2	$\pm 10ppm$	$\pm[5/32768]s$	$\pm[1/6] \cdot 10^{-3}s$	$\pm 2 \cdot 10^{-6}s$
s_3	$\pm 20ppm$	$\pm[5/32768]s$	$\pm[1/6] \cdot 10^{-3}s$	$\pm 2 \cdot 10^{-6}s$



Slika 2.6: Ocenjevanje maksimalne napake pri merjenju leteče hitrosti atleta.

Centralna enota se s prvim senzorjem časovno uskladi po preteku 0.5 s od zagona. Z naslednjimi senzorji se časovno uskladi z razmakom 1 s. Napaka ure centralne enote je pri vsakem usklajevanju večja zaradi večje oddaljenosti od referenčne časovne točke, kar je razvidno iz izračunov 2.3 po enačbi $N[ura_{ce}]_{sync_{s_x}} = (T_{sync_{s_x}} - T_{ref}) \cdot relN[ura_{ce}]$.

$$\begin{aligned} N[ura_{ce}]_{sync_{s_1}} &= (0.5s - 0.5s) \cdot \pm 2ppm = 0s \\ N[ura_{ce}]_{sync_{s_2}} &= (1.5s - 0.5s) \cdot \pm 2ppm = 1s \cdot \pm 2ppm = \pm 2 \cdot 10^{-6}s \\ N[ura_{ce}]_{sync_{s_3}} &= (2.5s - 0.5s) \cdot \pm 2ppm = 2s \cdot \pm 2ppm = \pm 4 \cdot 10^{-6}s \\ N[ura_{ce}]_{sync_{s_4}} &= (3.5s - 0.5s) \cdot \pm 2ppm = 3s \cdot \pm 2ppm = \pm 6 \cdot 10^{-6}s \end{aligned} \tag{2.3}$$

Več časa kot je poteklo od senzorjeve časovne uskladitve s centralno enoto,

večja je napaka ure, s katero uparjamo senzorjeve prekinitvene dogodke, kar je razvidno iz izračuna 2.4 po enačbi $N[ura_{s_x}]_{pr} = (T_{pr_{s_x}} - T_{sync_{s_x}}) * relN[ura_{s_x}]$.

$$\begin{aligned} N[ura_{s_0}]_{pr}1 &= (5s - 0.5s) \cdot \pm 2ppm = 4.5s \cdot \pm 2ppm = \pm 9 \cdot 10^{-6}s \\ N[ura_{s_1}]_{pr}2 &= (6s - 1.5s) \cdot \pm 7.5ppm = 4.5s \cdot \pm 7.5ppm = \pm 33.75 \cdot 10^{-6}s \\ N[ura_{s_2}]_{pr}3 &= (7s - 2.5s) \cdot \pm 10ppm = 4.5s \cdot \pm 10ppm = \pm 45 \cdot 10^{-6}s \\ N[ura_{s_3}]_{pr}3 &= (8s - 3.5s) \cdot \pm 20ppm = 4.5s \cdot \pm 20ppm = \pm 90 \cdot 10^{-6}s \end{aligned} \tag{2.4}$$

Napake prekinitev senzorjev po enačbi $N_{pr_s}x = N[ura_{ce}]_{sync_s} + N[sync_s] + N[ura_s]_{pr}x + N[hw_s]_{pr}x + N[sw_s]_{pr}x)$ so prikazane v izračunu 2.5.

$$\begin{aligned} N_{pr_{s_0}}1 &= \pm 0s + \pm[5/32768]s + N[ura_{s_0}]_{pr}1 + \pm[1/6] \cdot 10^{-3}s + \pm 2 \cdot 10^{-6}s \\ &\approx \pm 321 \cdot 10^{-6}s + N[ura_{s_0}]_{pr}1 \\ &\approx \pm 321 \cdot 10^{-6}s + \pm 9 \cdot 10^{-6}s \approx \pm 0.33 \cdot 10^{-3}s \\ N_{pr_{s_1}}2 &= \pm 2 \cdot 10^{-6}s + \pm[5/32768]s + \pm 33.75 \cdot 10^{-6}s + \pm[1/6] \cdot 10^{-3}s + \pm 2 \cdot 10^{-6}s \\ &\approx \pm 323 \cdot 10^{-6}s + \pm 33.75 \cdot 10^{-6}s \approx \pm 0.356 \cdot 10^{-3}s \\ N_{pr_{s_2}}3 &\approx \pm 325 \cdot 10^{-6}s + \pm 45 \cdot 10^{-6}s \approx \pm 0.370 \cdot 10^{-3}s \\ N_{pr_{s_3}}4 &\approx \pm 327 \cdot 10^{-6}s + \pm 90 \cdot 10^{-6}s \approx \pm 0.417 \cdot 10^{-3}s \end{aligned} \tag{2.5}$$

Maksimalna napaka meritve med dogodkoma prekinitev, ki sta se zgodila v časih $T_{pr_{s3}}3$ in $T_{pr_{s0}}0$ po neodvisni časovni komponenti je $\approx \pm 0.417 \cdot 10^{-3}s + \pm 0.33 \cdot 10^{-3}s \approx \pm 0.747 \cdot 10^{-3}s$. Če se čas med dogodki prekinitev povečuje, napaka narašča. Napaka je prav tako odvisna od oddaljenosti dogodkov prekinitev po času od referenčne točke T_{ref}. Napaka pri merjenju je manjša, če se prekinitve na senzorjih z večjo relativno napako ure zgodijo pred prekinitvami na senzorjih z manjšo napako ure. Da bi povečali natančnost meritve,

je potrebno časovno usklajevanje ponoviti tik pred meritvijo. To lahko storimo ročno oz. nam sistem omogoča avtomatizem, preko katerega se to izvaja implicitno na določenih časovnih intervalih, kar zavisi od implementacije sistema. Če je zadnja časovna uskladitev s senzorji z različnimi relativnimi napakami ur zelo oddaljena od začetnega časa meritve, so absolutne napake ur senzorjev ob prekinitvenih dogodkih lahko zelo velike. V skrajnem primeru je pod vprašajem vrstni red zabeleženih meritvenih dogodkov.

2.3.2 Pogoji delovanja

Pogoji, v katerih se izvajajo meritve, so lahko različni. Atleti tekmujejo pri temperaturah od 0 ℃ pa tja do okoli 40 ℃, ne glede na to ali dežuje ali piha močan veter. Največkrat se meritve hitrosti izvajajo na notranjih površinah, saj lahko tako zanemarimo spremenljive pogoje. Rezultate je mogoče efektivno primerjati ravno pri enakih pogojih izvedbe testov. Veter, ki piha atletu v hrbet z 1 m/s pripomore k rezultatu šprinta na 100 m za cca. 1/10 s. Rezultati na tekmah, ko pada dež, so praviloma slabši itn.

Upoštevati moramo napake v natančnosti sistema zaradi različnih pogojev, v katerih se lahko znajdejo posamezne enote. Sistem v osnovi ni primeren za merjenje daljših tekov, kjer so razdalje med centralno enoto ter senzorji prevelike. V tem primeru ni mogoče izvajati časovnih sinhronizacij ob poljubnem času, saj senzorji niso v radijskem dometu centralne enote.

Določiti je potrebno robne pogoje delovanja, v katerih sistem še deluje dovolj zanesljivo (temperaturno območje, vlažnost, frekvenca časovnih sinhronizacij ...).

2.3.3 Domet kominikacije in uporaba

Domet radijske komunikacije modula RFM69w je od 150 m do 200 m v idealnih pogojih. Težave lahko povzročajo komunikacijske motnje oz. odboji ter lomljenje signalov glede na prisotne fizične prepreke. Pri postavitvi senzorjev je potrebno stremeti k temu, da so le-ti v dometu centralne naprave, če je

le-to mogoče. S tem omogočimo centralni napravi, da izvaja časovne sinhronizacije, kadar je to potrebno. Trenerji večkrat razporedijo IR-vrata po tekmovalnih progah stadiona, sami pa se postavijo nekje na sredino zelenice, od koder opazujejo tekače. Največja razdalja med senzorjem in celico pa ne presega 100 m.

Rešitev lahko uporabljamo tudi pri drugih športih. Npr. na poligonu, ki predstavlja simulacijo začetka bob steze. Tu merimo čase med dogodki, ki predstavljajo prehode v točkah, ki jih bob siulator prevozi. Potem lahko merimo nogometaše na nogometnih površinah v času kondicijskih priprav itn.

Poglavje 3

Strojna oprema rešitve za merjenje časa

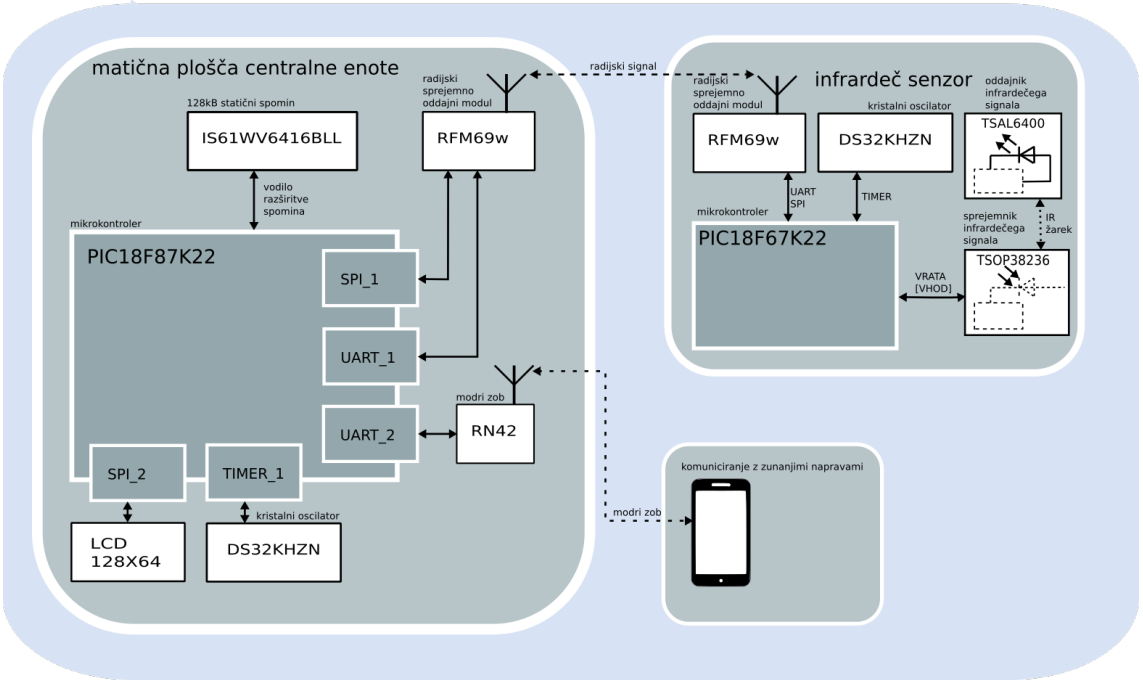
Strojna oprema meritvenega sistema sestoji iz centralne enote ter poljubnega števila senzorjev (slika 3.1). Centralna enota komunicira s senzorji v okviru funkcij oštevilčenja, časovnega usklajevanja ter izmenjave informacij o prekinitvenih dogodkih (podpoglavje 4.4). Centralna enota in senzorji predstavljajo lokacijsko distribuirane procesne enote. Osnovani so na centralno procesni enoti oz. mikrokontrolerju serije PIC18 proizvajalca Microchip. Mikrokontroler centralne enote ima razširjen pomnilni naslovni prostor z zunanjo statično pomnilno enoto IS61WV6416BLL, ki vsebuje 128kB. Vsaka enota ima svojo uro, ki jo poganja natančen ceneni kvarčni oscilator DS32KHZN s temperaturno kompenzacijo, ki niha pri 32768Hz. Med seboj usklajene ure predstavljajo relativen čas glede na zagon centralne enote.

Za komunikacijo med centralno enoto in senzorji se uporabljata: Uart modul mikrokontrolerja za asinhrono serijsko komunikacijo ter radijski modul RFM69w. Serijski podatki Uart modula se oddajajo/sprejemajo preko radijskega modula, ki deluje v načinu, ki omogača časovno determiniranost pošiljanja ter sprejemanja podatkov. Časovna determiniranost radio komunikacije je zahtevana pri usklajevanju relativnega časa med uro centralne enote ter urami senzorjev. Modul RFM69w konfiguriramo preko serijskega

podatkovnega modula SPI.

Centralna enota arhivira podatke o zajetih prekinitvenih dogodkih. Za hrambo podatkov uporablja mikrokontrolerjev interni električno izbrisljiv programirljiv bralni pomnilnik (EEPROM) velikosti 1kB. Za prikaz stanja programa uporablja enobarvni grafični LCD-ekran z matriko 128*64 točk. Interakcija uporabnika s sistemom je omogočena preko uporabe tipkovnice z matriko 8*2. Poleg tega je sistem dostopen zunanjim napravam preko Bluetooth komunikacije. Tako lahko naprave, kot so pametni telefoni, izvajajo akcije na centralni enoti ter prikazujejo njihove rezultate, če je enota v dometu bluetooth naprave.

Časovno usklajeni senzorji zajemajo prekinitvene dogodke v okviru trenutne meritve. Za zajem skrbijo sprejemno-oddajne enote infrardečih žarkov SFH5110, SFH4510.



Slika 3.1: Diagram strojne opreme — centralna enota, infrardeč senzor in povezava z zunanjimi napravami.

3.1 Centralna procesna enota

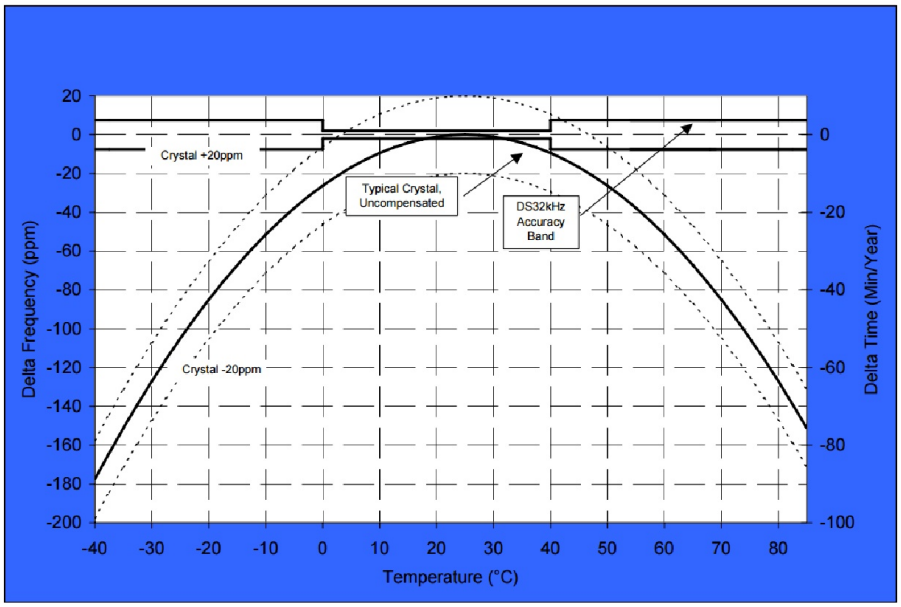
Jedra enot so mikrokontrolerji serije PIC18 proizvajalca Microchip. Uporaba temelji na kompromisu med ceno enote in njeno zmogljivostjo. Glavne značilnosti izbranega PIC18F87K22 mikrokontrolerja [1]:

- tehnologija s skrčenim naborom ukazov RISC,
- do 16 milionov ukazov na sekundo (MIPS),
- hitrosti ur do 64 MHz,
- napetostno območje delovanja : 1.8V to 5.5V,
- 128 kB internega bliskovnega programskega pomnilnika FLASH,
- 1kB EEPROM-pomnilnika,
- 4kB statičnega delovnega pomnilnika,
- 1,000,000 pisalno/brisalnih podatkovnih ciklov EEPROM-a z dobo pomnenja 40 let in več,
- temperaturni interval delovanja od -40 do 125 ℃.

Zahtevnost programske rešitve centralnega dela se je v zadnjem času tako zelo povečala, da najzmoglevejša različica te serije PIC18F87K22 z možnostjo razširjenega naslovnega pomnilnega prostora še komajda zadošča za delovanje. Trenutno je v pripravi testna različica matične plošče centralne enote z mikrokontrolerjem v ARM-tehnologiji.

3.2 Kvarčni oscilator

Kvarčni oscilator DS32KHZN [3] s temperaturno kompenzacijo deluje s frekvenco 32.768 kHz. Temperaturno območje delovanja je od -40 ℃ do 85 ℃.



Slika 3.2: Krivulja vrednosti frekvenčnega odstopanja kristala v odvisnosti od temperature, z metodo temperaturne kompenzacije in brez.[2]

Signal na izhodu je v obliki pravokotnega nihanja. Temperaturno kompenzacijo izvaja tako, da vsakih 64 s izmeri temperaturo in glede na to uravnava signal na izhodu.

S pomočjo temperaturne kompenzacije dosega natančnost signala na izhodu do ±2 ppm v temperaturnem območju od 0 ℃ do 40 ℃ ter ±7.5 ppm na območjih od -40 do 0 ℃ ter 40 ℃ do 85 ℃, kot je prikazano na sliki 3.2. Velikost napake v odvisnosti od različnih časovnih enot glede na temperaturno območje znaša od 0 ℃ do 40 ℃, prikazano v enačbi 3.1.

$$\begin{aligned} relN = \frac{2}{10^6} \cdot \frac{1s}{1s} &= \frac{2 \cdot 10^{-6}s}{1s} = \frac{2 \cdot 10^{-5}s}{10s} \\ &= \frac{12 \cdot 10^{-5}s}{1min} = \frac{7.2 \cdot 10^{-3}s}{1ur} = \frac{0.1728s}{1dan} \\ &= \frac{63.072s}{1leto} \end{aligned} \tag{3.1}$$

[2]slika skopirana iz specifikacije DS32KHZ modula [3] str. 5

Če privzamemo, da mora sistem delovati po kriteriju, v katerem je vsota vseh napak pri merjenju časa < 0.001 s, potem sleherna meritev ne sme trajati več kot 8.3 minute. Glede na to, da je rešitev namenjena predvsem za merjenje krajših razdalj, merjeni časi ne bodo presegli te vrednosti. V kolikor do tega pride, je potrebno uporabnika o tem nekako obvestiti preko uporabniškega vmesnika.

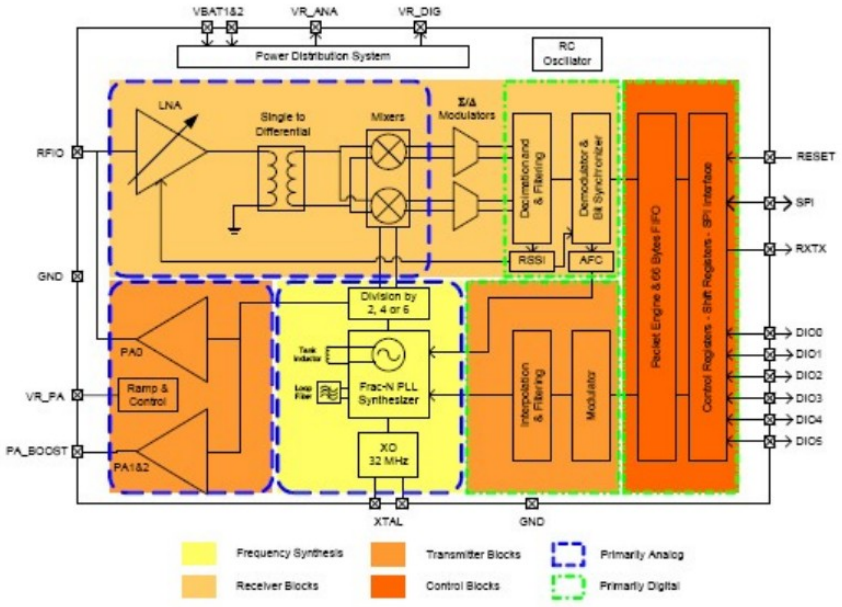
3.3 Radijski sprejemno/oddajni modul

HOPERF RFM69w [2] je radijski sprejemno/oddajni modul, ki je sposoben delovati na širokem frekvenčnem območju, vključno s frekvencami brez licenčnin 315, 433, 868 in 915 MHz (t. i. znanstveno industrijskem in medicinskem območju - ISM). Osnovni parametri radijsko frekvenčne komunikacije so programsko nastavljivi. Moč na izhodu dosega +13 dBm na napetostnem intervalu od 1.8 V do 3.6 V. Omogoča obojesmerno komunikacijo z izmeničnim oddajanjem (half-duplex). Modul konfiguriramo preko SPI-komunikacijskega vmesnika.

Osnova za lokalni oscilator je kontrolni sistem PLL (fazno zaključena zanka). Naloga zanke je sinhronizacija s frekvenco signala na vhodu ter generiranje stabilnega frekvenčnega signala. Generirana frekvenca je večkratnik frekvence signala na vhodu PLL-ja. Ta frekvenca je osnova za delovanje sprejemnega in oddajnega modula. Resolucija PLL-sistema je določena z enačbo $F_{STEP} = \frac{F_{XOSC}}{2^{19}} = \frac{32Mhz}{2^{19}}$, medtem ko nosilno frekvenco pri komunikaciji programsko določimo z: $F_{RF} = F_{STEP} * N$.

Pasovna širina kontrolnega sistema PLL omogoča velike hitrosti podatkovnih tokov do 300 kb/s ter hitro zaklepanje signala na vhodu, kar omogoča hitre zagonske čase ter hitro preskakovanje med frekvencami (kanali).

Oddajnik sestavljajo zgoraj omenjeni frekvenčni sintetizator, modulator ter bloki za močnostno ojačevanje signala. Modul omogoča modulacijo s frekvenčnim skokom FSK oz. modulacijo OOK (enostavna oblika modulacije z amplitudnim skokom). Le-to ter njene karakteristike lahko programsko kon-



Slika 3.3: Blok diagram radijskega sprejemno oddajnega modula RFM69w.[4]

figuriramo. FSK-modulacija se izvede s PLL-sistemom znotraj njegove pasovne širine. Gre za frekvenčno modulacijo, pri kateri je digitalna informacija kodirana preko diskretnih sprememb v frekvenci signala. OOK-modulacija se izvaja z vklapljanjem ter izklapljanjem ojačevalca moči signala.

Spektralno škropljenje lahko zmanjšamo in s tem izboljšamo odziv oddajnika na ožjem frekvenčnem pasu tako, da povečamo čas vklapljanja in izklapljanja ojačevalnikov moči.

V FSK-načinu lahko aktiviramo Gaussov filter, s katerim filtriramo moduliran signal. Pri načinu delovanja na podatkih v realnem času (continuous) se aktivira dodaten signal DCLK na nožici DIO1/DCLK, na kateri sproži prekinitev na mikrokontrolerju za vsak bit, ki ga je potrebno poslati.

Sprejemnik sestavljajo ojačevalnik z omejitvijo motenj (LNA), mikser, analogno digitalni pretvornik, kanalni filter, demodulator, indikator moči prejetega signala, bitni sinhronizator, avtomatska korekcija frekvence itn.

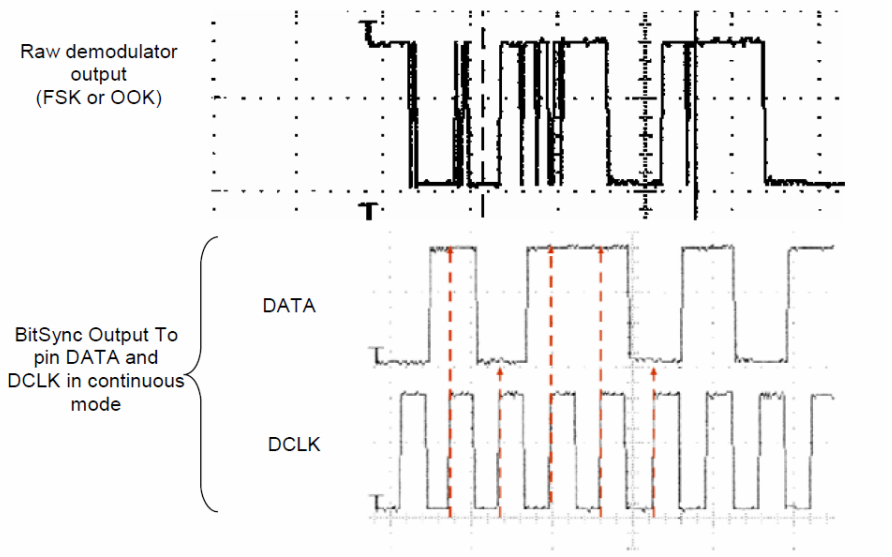
Filtriranje, demodulacija, kontrola ojačitve, sinhronizacija ter upravljanje

[4]slika skopirana iz specifikacije rfm69w modula [2] str. 8

s paketi se izvaja v digitalnem načinu.

Indikator moči prejetega signala RSSI predstavlja energijo znotraj pasovne širine kanala sprejemnika. Resolucija indikatorja je 0.5dB. Za izračun sta potrebni dve bit periodi. Vzorčenje pri FSK-modulaciji se mora zgoditi v času prejemanja uvodne besede (preamble). Pri OOK-modulaciji pa v času sprejemanja konstante vrednosti "1".

FSK-demodulator lahko demodulira FSK, GFSK, MSK in GMSK-signale. Demoduliran signal lahko nadalje pošljemo preko bitnega sinhronizatorja, ki ga očisti motenj ter zagotovi dodaten digitalni sinhronizacijski signal. Tega lahko uporabimo za vzorčenje vrednosti bitov podatkovnega signala (prikazano na sliki 3.4).



Slika 3.4: Izboljšanje signala z uporabo bitnega sinhronizatorja ter dodaten signal. [6]

Uporaba bitnega sinhronizatorja je zelo priporočena pri komunikaciji v realnem času (continuous). Za sinhronizacijo s signalom na sprejemu je potrebna uvodna beseda (0x55 ali 0xAA preamble) dolžine 12 bitov. Podatkovni tok, ki sledi mora imeti vsaj en prehod iz vrednosti '0' v '1' ali '1'

[6] slika skopirana iz specifikacije rfm69w modula [2] str. 31

v '0' vsakih 16 bitov med oddajo. Slednjemu pogoju zadostimo z uporabo Manchester kodiranja podatkovnega toka pred oddajo.

Avtomatska korekcija frekvence sloni na indikatorju frekvenčne napake, ki prikazuje razliko v frekvenci med lokalnim oscilatorjem ter nosilno frekvenco moduliranega signala na vhodu sprejemnika.

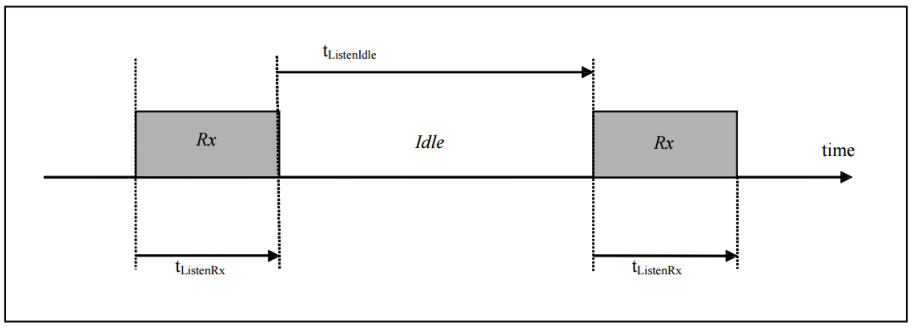
Modul podpira dva načina delovanja nad podatki:

- paketni način s FIFO-vrsto, kjer je vsak paket, poleg s podatkovno vsebino, opremljen še z uvodnim nizom (preamble), sinhronizacijsko besedo, CRC-izračunom itd.

- način v realnem času (continuous), kjer je vsak bit, poslan oz. prejet, dostopen v realnem času na nožici DIO2/DATA.

Slednji način uporabimo v naši rešitvi, in sicer brez uporabe Gausovega filtriranja. To nam omogoča, da na vhod oddajnika speljemo asinhron serijski podatkovni signal. To zagotovimo v kombinaciji z UART-modulom mikrokontrolerja. Sprejemni RX ter oddajni TX-nožici UART-mikrokontrolerjeve enote povežemo z DIO2/DATA-nožico modula RFM69w.

V določenem trenutku se modul nahaja v enem izmed 5 stanj: spanje (sleep), v čakanju (stand-by), sintetiziranje frekvence, oddajanje, sprejemanje oz. poslušanje (listen). Stanje v poslušanju je način, s katerim privarčujemo veliko energije. V bistvu gre za tip sprejemanja, kjer se sprejemnik intervalno vklaplja ter izklaplja, dokler se na radijski frekvenci ne pojavi signal, kot je prikazano na sliki 3.5. Prisotnost signala evaluira glede na vrednost indikatorja moči, in sicer ko ta prekorači določen prag.



Slika 3.5: Zaporednje dogodkov v stanju poslušanja. [7]

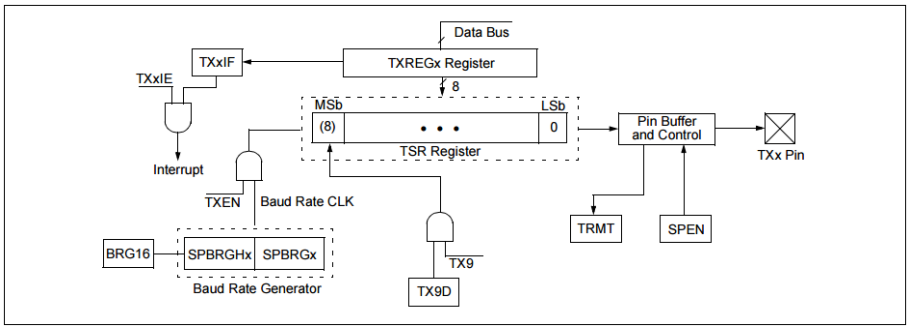
3.4 Modul za asinhrono serijsko komunikacijo

UART-enoto nastavimo v asinhronem načinu delovanja. V asinhronem načinu za delovanje skrbijo generator ritma (baudrate), asinhroni sprejemnik in asinhroni oddajnik. Za kodiranje podatkov se uporablja standardni NRZ (Non-Return-to-Zero) format (en Start bit, osem ali devet podatkovnih bitov ter en Stop bit). Najpogosteje uporabljen podatkovni format je 8 bitov. 8-bitni/16-bitni generator ritma omogoča standardne frekvence (1200, 4800, 9600, 19200 ...) na osnovi oscilatorja.

Diagram oddajnika je prikazan na sliki 3.6. Srce oddajnika je pomikalni register (TSR). Ta podatke dobi iz oddajnega vmesnega pomnilnika (TXREG), ki ga polnimo iz programske kode. Ko je odposlan STOP bit iz prejšnjega polnenja, se TSR ponovno napolni z vsebino TXREG-a. Mikrokontroler obvesti uporabnika, da se je TXREG izpraznil, s prekinitvijo na predkonfigurirani prioritetni niti. V tem primeru mora uporabnik bodisi ponovno napolniti TXREG z novim bajtom, in s tem deaktivirati prekinitev, bodisi prekiniti pošiljanje. Pomikalni register se prazni z baudno hitrostjo. Vrednost pomaknjenega bita se odrazi na izhodu TX.

Diagram sprejemnika je prikazan na sliki 3.7. Sprejemnik podatke zajema na vhodu RX. Zajem se vrši preko zelo hitrega pomikalnega registra, ki deluje

[7] slika skopirana iz specifikacije rfm69w modula [2] str. 39



Slika 3.6: Diagram uart oddajnika. [8]

s 16-kratnikom baudne hitrosti, medtem ko se glavni bralni pomikalni register polni z baudno hitrostjo.

Ko pride do pravilnega sprejetja bajta, se ta pojavi v pomnilnem registru RCREG. Mikrokontroler posledično izvede prekinitev na predkonfigurirani prioritetni niti. Programska koda uporabnika mora podatek prebrati iz RCREG-registra in s tem deaktivirati prekinitev oz. prekiniti sprejemanje podatkov. Če tega ne uspemo napraviti v času prejemanja naslednjega bajta, lahko pride do napake prekoračitve podatkov.

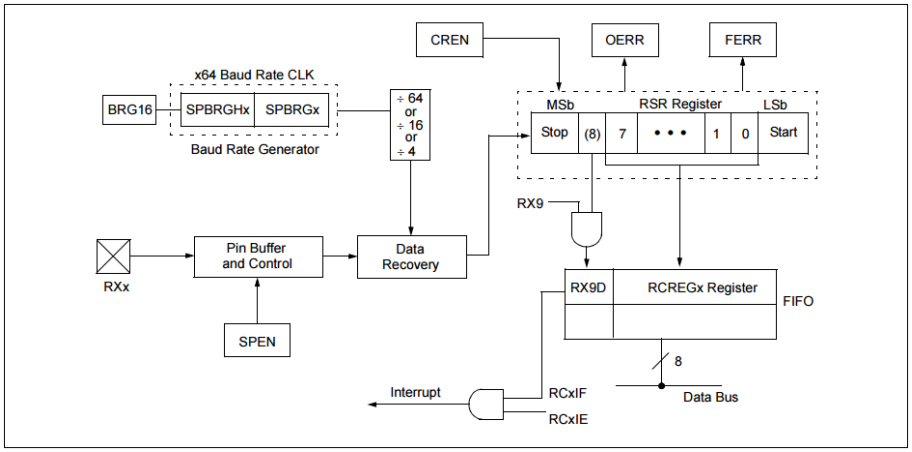
V kolikor pride do napake pri sprejemanju meta podatkov (neporavnani START, STOP biti), se to zavede kot napaka pri branju okvirja. Omogočeno je pošiljanje 12-bitnega prekinitvenega (break) znaka, zbujanje iz spalnega načina ter zaznavanje in korekcija ritma ob sprejemu sinhronizacijskega prekinitvenega znaka.

Pri časovni sinhronizaciji pričakujemo deterministično vrednost časa, ki je potreben za prenos podatkovnega paketa iz enote A v enoto B. V primeru uart modula gre za vrednost, ki preteče od trenutka, ko napolnimo TXREG-enote A s prvim bajtom sinhronizacijskega paketa, do trenutka, ko v enoti B prejmemo prekinitev, ki nakazuje prejem zadnjega bajta istega paketa.

Skupna napaka pri pošiljanju podatkov z uporabo modulov Uart in rfm69w sestoji iz:

- Uart napake pri determiniranosti časa od začetka oddaje (vnos po-

[8] slika skopirana iz specifikacije uC PIC18F87K22 [1] str. 338



Slika 3.7: Diagram uart sprejemnika. [9]

datka v TXREG) do signala na TX-nožici, ki nakazuje začetek okvirja (START bit). Ta napaka je odvisna od stabilnosti frekvence signala, ki poganja mikrokontroler in je zanemarljiva (sploh zaradi dejstva, da se jo upošteva enkrat na dogodek sinhronizacije).

- napaka zakasnitve, ki je potrebna pri pošiljanju signala preko RF modula rfm69w.

- Uart napaka na sprejemu pri zaklepanju na START bit (začetek okvirja).

Slednjo napako pri zaklepanju na START bit lahko zgoraj omejimo na $< 0.5 \cdot \frac{1}{BAUDRATE} s$. Uart sprejemnik preko začetka okvirja določi točke, v katerih se vzorčijo podatkovni biti. V idealnem primeru je to na sredini vsakega sprejetega bita. Če točko vzorčenja premaknemo za količino znotraj intervala napake, je vzorčenje še vedno uspešno. Pri standardni baudni hitrosti 9600 je napaka cca. $< 5 \cdot 10^{-5} s$.

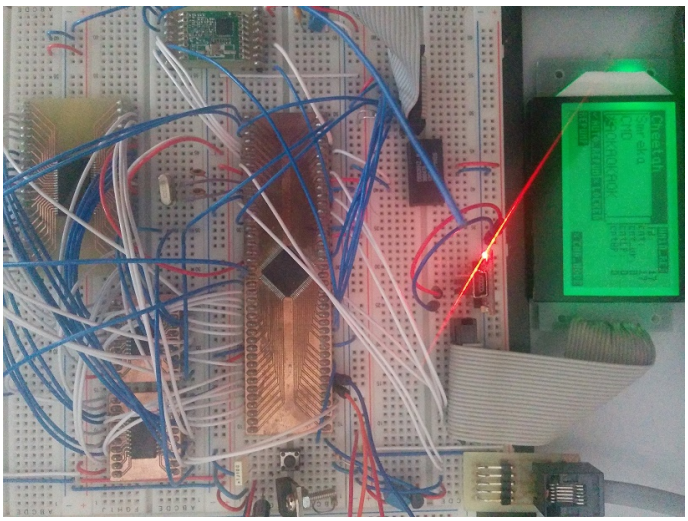
Sicer pa je napaka še manjša, saj vzorčenje START bita poteka s frekvenco, ki je 16-kratnik baudne hitrosti. Torej pri baudni hitrosti 9600 je napaka manjša od $\frac{1}{9600*16} s$. V vsakem primeru pa je napaka pri Uart sprejemu odvisna od baudne hitrosti.

[9] slika skopirana iz specifikacije uC PIC18F87K22 [1] str. 340

3.5 Centralna enota

Matično ploščo centralne enote sestavljajo:

- mikrokontroler PIC18F87K22 (3.1) z zunajno razširitvijo statičnega rama 128K bajtov,
- DS32KHZN kvarčni oscilator s temperaturno kompenzacijo (TCXO) kot izvor urinega takta (3.2),
- HOPERF RFM69w radijski sprejemno/oddajni modul (3.3),
- Roving RN42 bluetooth modul (3.5.1),
- TG12864H3 monokromatski LCD zaslon z matriko 128 X 64 točk,
- testna tipkovnica z matriko 2 X 4 (8 tipk).

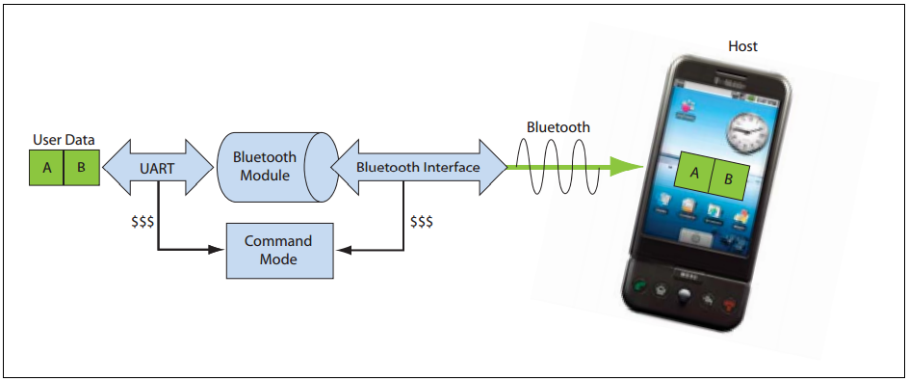


Slika 3.8: Matična plošča centralne enote v testni fazi.

Slika 3.8 prikazuje prototip centralne enote. Elektronsko vezje je implementirano na testni plošči (protoboard), ki omogoča povezovanje kontaktov z žicami. Za vsa integrirana vezja (čipe) se uporabljajo posebej prirejeni adapterji, preko katerih jih lahko priključimo na testno ploščo.

3.5.1 RN42 bluetooth

RN42 [4] je modul, ki podpira bluetooth (BT) protokol različice 2. Modul se upravlja z ASCII ukazi preko UART-komunikacijskih vrat. Mikrokontroler tako konfigurira lastnosti modula, prebira statusne spremenljivke ter upravlja s konekcijami z drugimi BT-moduli.



Slika 3.9: Podatkovni in ukazni način RN42 modula. [10]

Modul lahko deluje v podatkovnem oz. v ukaznem načinu [5]. V podatkovnem načinu modul prenaša podatke, ki jih pošljemo preko UART-vrat, povezanemu modulu na drugi strani bluetooth konekcije (če le-ta obstaja) in obratno. V ukaznem načinu modul izvaja ukaze ter na UART-vrata vrača povratne informacije teh ukazov (slika 3.9). Primeri ukazov: $$$ (vstopi v ukazni način), — (izstopi iz ukaznega načina), SN,<string> (nastavi ime), R,1 (ponovni zagon).

Ima več načinov delovanja. Naša rešitev uporablja način SUŽENJ, v katerem lahko druge naprave odkrivajo in se povežejo z modulom.

Modri zob modul uporabljamo za dostop zunanjih naprav do naše centralne enote. Aplikacija, ki teče na centralni enoti, objavlja del aplikacijskega programskega vmesnika. Tako lahko preko pametnih telefonov izvajamo funkcije na centralni enoti in ob tem prejemamo povratne podatke.

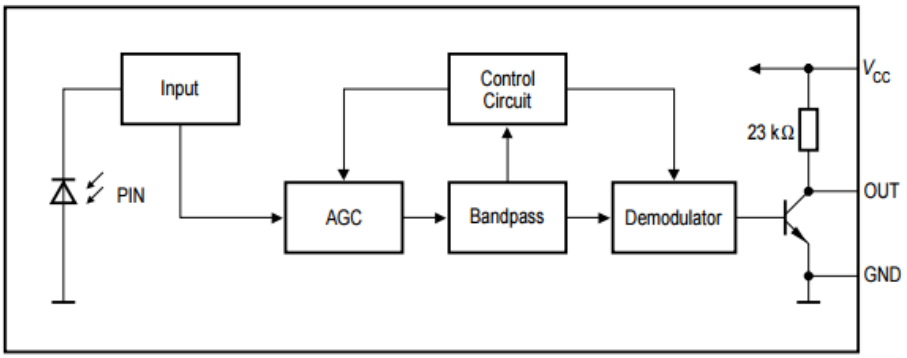
[10] slika skopirana iz specifikacije ukaznega načina modula RN42 [5] str. 6

3.6 Senzorji

Matična plošča senzorjev je v večji meri enaka plošči centralne enote. Osnova je enak oz. manj zmogljiv mikrokontroler serije PIC18. Za urin takt skrbi enak kvarčni oscilator ter za komunikacijo s centralno enoto enak radio sprejemno/oddajni modul. Za razliko od centralne enote ne podpira bluetooth komunikacije, saj odgovarja le centralni enoti. Imamo dva tipa senzorja:

- vrata z infrardečim (IR) žarkom in
- stikalo v obliki traku.

3.6.1 Sprejemnik in oddajnik IR-žarka



Slika 3.10: Diagram elementov integriranega vezja SFH5110. [11]

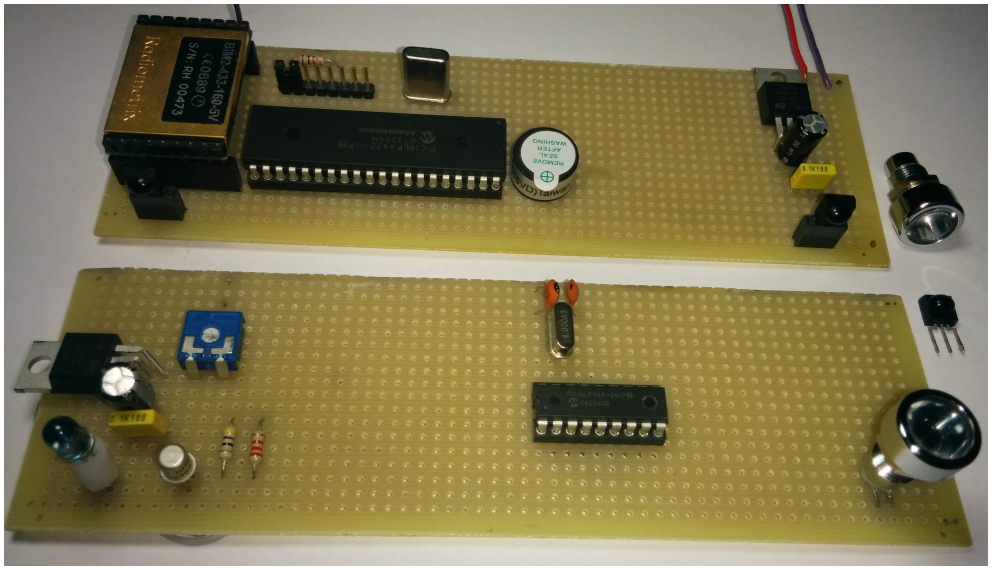
Pri vratih z IR-žarkom mikrokontroler komunicira z IR-sprejemmikom. Gre za modul tipa SFH5110 [6] podjetja Osram, ki je sestavljen iz foto detektorja svetlobe valovne dolžine 940 nm, predojačevalca ter filtra za impulzno-kodno modulacijo (PCM) (slika 3.10).

Kadar signal-šum razmerje analognega signala pada zaradi dušenja, se pri ojačanju tega signala ojača tudi šum. Digitalne signale je lažje ločiti od šuma in tako predstaviti v prvotni obliki. Zato moramo signal, ki ga oddajamo preko oddajne infrardeče diode, impulzno-kodno modulirati. Signal

[11] slika skopirana iz specifikacije modula SFH5110 [6] str. 4

predstavimo v obliki pravokotnega nihanja s preddoločeno frekvenco, ki je v primeru SFH5110-36 modula enaka 36 kHz. Modul je visoko imun na ambientno svetlobo. Minimalna širina podatkovnega signala za 1 bit je $\frac{6}{F_0}$ pri čemer F_0 predstavlja nosilno frekvenco (PCM). Modul omogoča neprekinjen signal do $\frac{F_0}{6}$ bps (bitov/s), kar znaša pri nosilni frekvenci 36 KHz 6000 bps. Iz tega sledi, da omogoča natančnost pri merjenju prekinitev do $\frac{1}{6000}s$.

Maksimalna razdalja med oddajnikom tipa SFH 4510/SFH 4515 [7] (pri IF = 500 mA) in sprejemnikom je okoli 30 m. Moč oddajne diode je potrebno prilagoditi razdalji z namenom zmanjšanja porabe energije. Zmanjšamo lahko tudi obratovalni faktor podatkovnega signala v takšni meri, da še vedno zadostuje pogoju za natančnost časovne prekinitve (napaka $< \frac{1}{1000}$s). Valovna dolžina infrardeče svetlobe je 940 nm. Izhod modula, ki ga priključimo na mikrokontroler je aktiven pri nizki vrednosti. Uporabimo lahko dodatne leče za fokusiranje IR-diod, kar nekoliko poveča domet svetlobe pri enaki porabi energije (slika 3.11).



Slika 3.11: Matična plošča senzorja z IR-sprejemnikoma in plošča z IR-oddajnima diodama. Prikazana starejša različica z radijskim sprejemno oddajnim modulom BIM2 podjetja Radiometrix

Mikrokontroler, ki kontrolira oddajni signal, mora poskrbeti za impulzno-

kodno modulacijo. IR-oddajnik lahko predstavlja samostojno enoto ali pa je skupaj s sprejemnikom nameščen na senzorski matični plošči. V slednjem primeru potrebujemo odsevnik, v katerega usmerimo oddajno ter sprejemno diodo.

Pri uporabi dveh ali več žarkov, razporejenih po vertikalni črti, moramo izhode IR-sprejemnikov povezati z NOR-operacijo. Namreč, da se zavede dogodek prekinitve, mora priti do prekinitev vseh žarkov v vertikali.

Senzor je opremljen tudi s piezo zvočnim javjalnikom za zvočno nakazovanje dogodkov.

Poglavje 4

Programska oprema

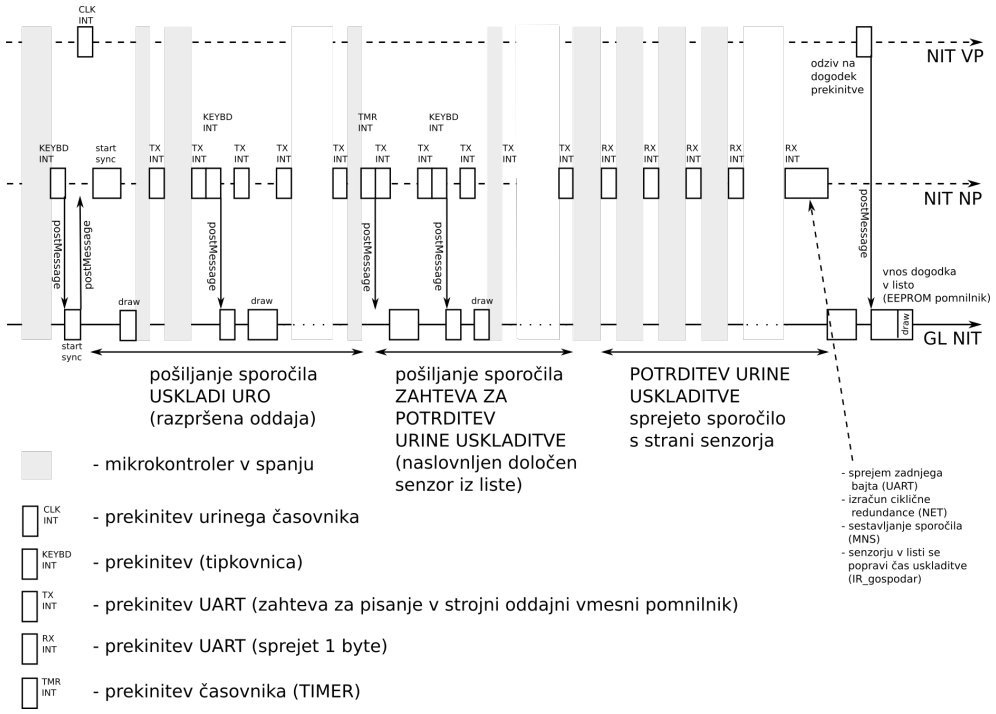
V prvem delu poglavja izpostavimo najpomembnejše programske module, ki so del operacijskega sistema.

Osrednji modul, t. i. sistem prenašanja sporočil, omogoča izvajanje funkcij objektov na različnih prioritetnih nitih. Ta nam omogoča objektno orientiran model programiranja rešitve. Domeno problema lahko opredelimo z objekti in njihovimi interakcijami, implementiramo lahko hierarhijo dedovanja med tipi objektov itn.

Funkcije kličemo preko pošiljanja sporočil naslovljenim ciljnim objektom. Sporočila se sortirajo v prioritetnih vrstah, ki pripadajo prioritetnim nitim, na katerih se funkcionalnosti izvajajo. S porazdeljevanjem nalog po prioritetnih nitih lahko povečamo odzivnost sistema. S tem posledično zmanjšamo napake, kjer časovne deterministočnosti odzivov na dogodke ni mogoče doseči.

V trenutni implementaciji sistem prenašanja sporočil zajema tri niti, in sicer glavno programsko nit ter dve prekinitveni niti z višjimi prioritetami (nit z nizko prioriteto ter nit z visoko prioriteto), kot je razvidno na sliki 4.1.

Na glavni programski niti se izvaja programska zanka, ki bere sporočila iz prioritetne vrste. Prebrana sporočila zanka sproti razpošilja ciljnim objektom, v okviru katerih se kot odziv izvedejo določene akcije. Ko glavna zanka iz vrste prebere zadnje sporočilo ter ga pošlje ciljnemu objektu, se izvede risanje grafike. Risanje grafike ima tako najnižjo prioriteto v sistemu. Zanka



Slika 4.1: Procesiranje dogodkov na prioritetnih nitih (glavna nit, nit nizke ter nit visoke prioritete).

nato obstoji pri branju iz prazne vrste (nit gre v spanje), vse dokler se ne pojavi novo sporočilo. Nato se postopek ponovi.

Višje prioritetne niti postanejo aktivne ob bodisi programskih bodisi strojnih prekinitvah. Prekinitvena nit izvede prekinitvene odzivne rutine in nato pri pogoju, da so vse prekinitve obdelane, ponikne (preide v neaktivno stanje) ter s tem prepusti procesni čas nitim z nižjo prioriteto, sicer pa se ponovno izvede. Ob aktiviranju prekinitvene niti se prvo izvede zanka, ki bere ter razpošilja sporočila iz prioritetne vrste, ki ji pripada. Ko se dotična vrsta izprazni, nit nadaljuje z izvajanjem prekinitvenih odzivnih rutin. Vsaki prioritetni niti pripada svoja prioritetna vrsta.

Del funkcionalnosti določenih objektov različnih programskih modulov predstavimo na tak način, da lahko izvajamo njih funkcije preko pošiljanja sporočil. Tako lahko pošiljamo sporočila nitim različnih prioritet v pripadajoče vrste. Če pošljemo sporočila v vrste prekinitvenih niti, je potrebno te

niti tudi aktivirati. To storimo s programsko prekinitvijo. Zanka na začetku aktivne prekinitvene niti nato razpošlje sporočila ciljnim objektom.

Funkcionalnosti objektov komunikacijskega sklada pri sprejemanju podatkov s strani lokacijsko distribuiranih enot se izvajajo na niti z nizko prioriteto (slika 4.1). V primeru sprejema podatkov o prekinitvenem dogodku meritve se na izvajajoči niti formira sporočilo s prejeto informacijo, ki se ga nato pošlje na obdelavo na glavno programsko nit z nižjo prioriteto. Ko pride sporočilo na vrsto, ga glavna zanka razpošlje objektu, ki poskrbi, da se dogodek meritve, kot informacija poslanega sporočila, vstavi v listo dogodkov trenutno aktivne meritve. Vnos meritvenega dogodka v listo, ki se nahaja v počasnem EEPROM-pomnilniku je procesno zahtevna operacija, zato je ni smotrno izvesti na niti nizke prioritete, saj bi s tem zmanjšali odzivnost niti na prekinitve.

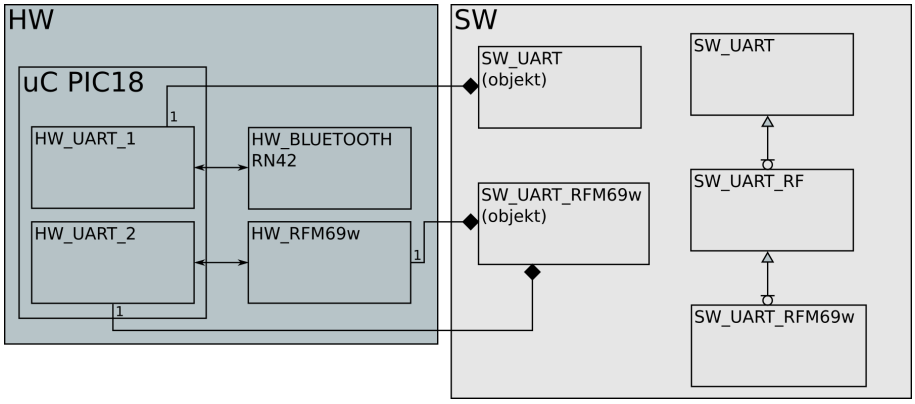
Zunanje naprave zaradi časovne nedeterminiranosti Bluetooth komunikacije niso časovno usklajene s centralno enoto (kot pri senzorjih). Zaradi tega lahko vse funkcionalnosti sprejemanja ukazov ter podatkov izvajamo na glavni niti, ki ima najnižjo prioriteto. Odzivna rutina hw_Uart modula, ki je del Uart objekta sprejete podatke vstavlja v vmesni pomnilnik. Sprejeti podatki se obdelajo na niti z najnižjo prioriteto. Odzivna rutina obvesti o sprejetju podatkov s poslanim sporočilom.

Tu je modul za upravljanje s časom, katerega naloga je v vsakem trenutku zagotoviti lokalno uro, ki predstavlja relativen čas glede na zagon centralne enote. Omogoča tudi ponastavljanje trenutne vrednosti ure za namene časovnega usklajevanja. Lokalno uro uporabljamo za uparjanje prekinitvenih in sinhronizacijskih dogodkov ter v druge namene, kot npr. za uporabo pri implementaciji časovnikov (timer) itn.

Potem imamo komunikacijski sklad, sestavljen iz: modula za asinhrono serijsko kominikacijo, modula za upravljanje paketov ter modula za pošiljanje in sprejemanje sporočil.

Objekt za asinhrono serijsko kominikacijo tipa Uart je gonilnik operacijskega sistema, ki upravlja z HW_Uart_x modulom mikrokontrolerja ter poleg

tega v radijski različici še z modulom rfm69w. Za vsak strojni modul se napravi svoj Uart oz. UartRF programski objekt. UartRF je razširitev tipa Uart.



Slika 4.2: Entitetni in objektni diagram, ki prikazuje strukturo dedovanja Uart tipov ter povezavo med Uart programskimi gonilniki in strojnimi hwUart ter rfm69w objekti.

Objekt omogoča uporabo vmesnih pomnilnikov, sprejemanje/oddajanje podatkov na različnih prioritetnih nitih, potem v radijski različici uporabo manchester kodiranja podatkov, pošiljanje uvodnih podatkov (preamble), sprejemanje z intervalnim poslušanjem itn.

Bluetooth modul RN42 je povezan z HW_Uart modulom mikrokontrolerja. Za komunikacijo uporabimo gonilnik tipa Uart. Medtem, ko pri komunikaciji preko rfm69w uporabimo radijski gonilnik tipa UartRF_rfm69w, ki je izpeljan iz UartRF. Ta gonilnik upravlja s strojnim HW_Uart_x modulom ter modulom rfm69w. Serijski podatki strojnega HW_Uart_x modula se pošiljajo/sprejemajo preko rfm69w. Gonilnik omogoča uporabo manchester kodiranja ter pošiljanje uvodnih (preamble) podatkov. RFM69w modul sicer sam omogoča manchester kodiranje ter pošiljanje uvodnih podatkov s strojno logiko, vendar to le v načinu za upravljanje s paketi, katerega v naši rešitvi ne uporabljamo. Paketni način modula namreč ne zagotavlja pošiljanja/sprejemanja podatkov v realnem času. Povezave med objekti prikazane na sliki 4.2.

Naslednji v skladu je modul za upravljanje s paketi. Ta vsebino podatkov dodatno opremi z metapodatki in tako formira pakete. Metapodatki predstavljajo velikost paketa ter ciklično redundančno kodo CRC velikosti dveh bajtov. Podatke paketov pošilja/sprejema preko Uart modula. Pri sprejemanju paketov sproti preverja velikost ter ciklično redundanco. V kolikor se ta dva ujemata, modul naznani sprejetje paketa.

Zadnji v skladu je modul za pošiljanje/sprejemanje sporočil. S tem modulom lahko pošiljamo sporočila med lokacijsko distribuiranimi enotami na podoben način, kakor to storimo z modulom sistem prenašanja sporočil lokalno. Vsaki enoti za sprejem ter oddajo sporočil pripada svoj lokacijski ident. Sporočila lahko naslavljamo posameznim enotam oz. z razpršeno oddajo vsem, ki poslušajo. Pri oddajanju se sporočila pretvorijo v serijsko obliko, ki predstavlja podatke paketov, ki se obdelajo v paketnemu modulu. Obratno se pri sprejemu podatki paketov pretvorijo iz serijske oblike v sporočila. Modul omogoča funkcionalnost odmev (ECHO). Naslovljeni enoti pošljemo sporočilo tipa ECHO. Ko naslovljena enota sporočilo prejme, ga takoj vrne pošiljatelju. S to funkcionalnostjo pošiljatelj okvirno izmeri čas trajanja pošiljanja sporočila naslovniku. To vrednost potrebujemo pri usklajevanju relativnega časa med centralno enoto ter senzorji.

V okvir operacijskega sistema spadajo tudi drugi pomembni gradniki:

- modul z virtualnimi časovniki (virtual timers). To je gonilnik, ki omogoča uporabo večih virtualnih časovnikov naenkrat nad ožjo množico strojnih časovnikov, ki so del mikrokontrolerja. Deluje s časovno resolucijo 1 ms. Podpira izvajanje opravilnih rutin na različnih prioritetnih nitih.

- sinhronizacijskimi elementi za kontrolo dostopa do sekcij, do katerih se dostopa z različnih niti; V primeru, da več niti dostopa do istega vira, z namenom pisanja in branja, lahko prihaja do nepravilnosti v prebranih podatkih. V tem primeru moramo zakleniti dostop do teh sekcij. To storimo preko elementov za ekskluzivno zaseganje. Gre za programski objekt, ki dovoljuje večim programskim nitim souporabo istega vira podatkov, v kolikor ne gre za hkraten dostop. V sistemu

implementiramo dva tipa objektov za ekskluzivno zaseganje, in sicer eksluzivno zaseganje za branje in pisanje ter eskluzivno zaseganje za pisanje (več bralcev oz. en pisalec naenkrat).

- grafični modul s/z: različnimi kontrolami, kot so oznaka (label), preglednica (datagrid), potrditveno polje (checkbox), itn.; podporo fontov različnih velikosti, raznimi funkcijami za risanje znakov, znakovnih nizov, geometrijskih teles itd.

- modul avtomat stanj, ki omogoča s stanji orientirano programiranje. Gre za matematični model, ki je pogleg drugih namenov uporabljen za dizajniranje računalniških programov. Avtomat se v določenem trenutnu nahaja v enem izmed končnega števila stanj (t. i. trenutno stanje). Avtomat lahko prehaja iz enega stanja v drugo. To se zgodi na podlagi dogodkov, ki so naslovljeni na avtomat. Odziv avtomata na dogodek je odvisen od narave dogodka ter trenutnega stanja avtomata. Implementiran avtomat je hierarhične narave, kjer lahko stanja gnezdimo. Določeno stanje lahko podeduje funkcionalnosti stanja, v katerem je vgnezdeno. Funkcionalnost je uporabljena v programski kodi senzorja. Objekt IR_suženj predstavlja avtomat s tremi stanji v hierarhiji. To so stanja "seja", "povezan in časovno neusklajen" ter "časovno usklajen". Vsako od teh stanj ima svoje funkcionalnosti, katere se izvajajo preko dogodkov s strani tipkovnice, komunikacijskega sklada ter časovnikov. Preko komunikacijskega sklada v določenem trenutku prejmemo dogodek "uskladitev ure", na katerega avtomat odreagira le v stanju "povezan in časovno neusklajen" ter njegovem podstanju "časovno usklajen". Tu se funkcionalnost, ki obravnava omenjeni dogodek, podeduje v podstanje. Če je uskladitev ure uspešna, bodisi ostanemo v stanju "časovno usklajen" bodisi v to stanje preidemo iz stanja "povezan" in "časovno neusklajen".

- modul za dinamično dodeljevanje pomnilnika, posebej prirejen za sisteme z omejeno velikostjo statičnega pomnilnika. Modul ob zagonu

aplikacije rezervira del statičnega pomnilnika, iz katerega na zahtevo uporabnika dodeljuje segmente bajtov poljubnih velikosti s funkcijo "malloc". Ko določenen segment bajtov ni več potreben, ga lahko preko funkcije "free" sprostimo in je tako na voljo za nadaljna dodeljevanja. Omejitev modula je, da lahko v trenutku rezerviramo ne več kot 127 bajtov, če je velikost bazena, iz katerega modul dodeljuje bajte večja oz. enaka velikosti 127 bajtov in hkrati, da obstaja takšna velikost prostih bajtov v enem kosu še nedodeljena. Razlog je prostorska optimizacija modula.

- gonilniki za LCD-zaslone z matriko 128x64 z vmesnim zaslonskim pomnilnikom.

- gonilnik za tipkovnico z matriko 8x2.

Drugi del poglavja opisuje samo programsko rešitev za merjenje časa. Pri programiranju je uporabljena paradigma model, pogled, krmilnik (MVC).

MVC-paradigma narekuje, da pri razvoju programske opreme uporabimo tri glavne komponente, kot so model, ki predstavlja logično strukturo podatkov aplikacije, pogled, ki predstavlja grafične objekte uporabniškega vmesnika ter krmilnik, ki povezuje model in pogled ter predstavlja njihovo komuniciranje.

Arhitekturo programske rešitve določajo sledeči gradniki:

- kronometer (timing), ki implementira vmesnik s funkcijami start za začetek meritve, lap za vmesne čase ter stop za konec meritve. Za hranjenje podatkov o meritvah se uporablja modul "pomnilnik meritev".

- modul "pomnilnik meritev" omogoča hranjenje podatkov o meritvah po posameznih atletih. Merimo lahko več atletov, ki opravijo poljubno število meritev, ki sestojijo iz dveh ali več prekinitvenih dogodkov (vsaj dva dogodka sta potrebna za meritev, in sicer štartni in končni dogodek). Podatki se hranijo v EEPROM-pomnilniku, katerega omejitev je 1 kB.

- modul centralne enote IR-gospodar (IR-master), katerega naloga je oštevilčenje senzorjev, usklajevanje ur senzorjev z uro centralne enote ter prejemanje prekinitev senzorjev in apliciranje le-teh v modul kronometer. Pri komunikaciji s senzorji uporablja protokol za izogibanje trkov. IR-gospodar hrani listo trenutno oštevilčenih senzojev, kjer je vsak senzor uparjen s časom dogodka sinhronizacije časa.

- modul senzorja IR-suženj (IR-slave), ki je zadolžen za komuniciranje z IR-gospodarjem. Nahaja se lahko v treh stanjih: nepovezan, povezan, povezan in časovno sinhroniziran. V stanju "časovno sinhroniziran" prejema prekinitvene dogodke od IR-sprejemnega modula, ki jih uparja s trenutnim časovnim markerjem. Dogodke naknadno vstavlja v vrsto za pošiljanje gospodarju. Dogodek se briše iz vrste v trenutku potrditve gospodarja, da je prekinitveno sporočilo prejel.

- grafična komponenta, ki skrbi za grafični pogled po MVC-paradigmi. Povezuje se z zgoraj omenjenimi krmilnimi komponentami, ki preko dogodkov krmilijo grafični prikaz. Uporabnik komunicira z vmesnikom preko tipkovnice, ki pošilja dogodke sprememb stanj tipkovnice krmilnim komponentam.

4.1 Modul za upravljanje s časom

Naloga modula za upravljanje s časom je zagotoviti uporabniku dostop do relativnega časa, ki predstavlja zagonski čas enote. Ko so ure enot usklajene, predstavlja to zagonski čas centralne enote.

Deluje v povezavi s strojnim modulom mikrokontrolerja po imenu časovnik. Časovnik poganja bodisi notranji oz. zunanji izvor nihanja z določeno frekvenco. Njegova naloga je štetje nihajev in obveščanje uporabnika v primeru, da je število preseglo določeno vrednost. Uporabnika obvesti s programsko prekinitvijo na najvišji prioriteti.

V trenutni izvedbi časovnik poganja zunanji izvor, in sicer gre za osci-

lator, ki niha s frekvenco 32768 Hz. Časovnik ima 2 bajta pomnilnika, v katerem hrani trenutno število nihajev. Do prekinitve pride vsaki 2 sekundi, ko časovnik prešteje do vrednosti 2^{16}. Z uporabo prekinitve omogočimo razširitev shrambe za vrednost števca na 4 bajte, in sicer dvema bajtoma strojnega dodamo 2 bajta programskega pomnilnika. To zadošča za cca. 36 ur delovanja.

Listing 4.1: Programski aplikacijski vmesnik časovnega modula.

```
1 ULONG getTimeStamp(void);
2 typedef struct time {
3    UWORD thousandsOfASecond;
4    BYTE seconds, minutes, hours;
5 } Time;
6 Time timeFromTimeStamp(ULONG);
7 ULONG timeStampFromTime(Time);
```

Kot glavna funkcija programskega vmesnika modula nastopa "getTimeStamp()" (vrstica 1 izseka 4.1). Rezultat klica funkcije je podatek velikosti 4 bajte, ki predstavlja zgoraj omenjeno število nihajev oscilatorja, ki jih je do sedaj preštel časovnik.

Funkcijo "getTimeStamp" lahko kličemo z niti različnih prioritet. Razširjena vrednost števca pa se povečuje na niti z najvišjo prioriteto. Z branjem strojne 16-bitne vrednosti časovnika nimamo problema, saj nam uC zagotavlja atomarno branje. Drugače je pri razširitvi števca z dvema bajtoma glede na to, da imamo opravka z 8-bitno različico mikrokontrolerja. Funkcija mora prebrati 16-bitno strojno vrednost števca in dvakrat po en razširitveni bajt, da lahko sestavi trenutni časovni marker. Med prebiranjem vseh treh segmentov se lahko zgodi prekinitev, ki jo časovnik aktivira ob dogodku, ko je vrednost strojnega števca presegla maksimalno vrednost (z 0xffff na 0x00 v heksadecimalni obliki). V tem primeru lahko pride do napake pri izračunu končne vrednosti markerja. Zato moramo tu dodati logiko, ki preverja, ali se je pri določenem segmentu branja zgodila prekinitev in temu izračun primerno

prilagoditi.

Vmesnik nudi tudi pomožne funkcije za preračun časovnega markerja v standardne časovne enote (sekunda, minuta ...) in obratno, kot je razvidno iz izseka kode 4.1.

4.2 Sistem prenašanja sporočil

Ideja programskega modula je omogočiti izvajanje funkcij objektov na različnih prioritetnih nitih. Rešitev je implementirana direktno nad funkcionalnostmi mikrokontrolerja brez vmesnega operacijskega sistema. Pri razvoju so bili uporabljeni mikrokontrolerji serije PIC18 znanega proizvajalca Microchip. Enota izvaja programsko kodo na glavni in dveh prioritetnih nitih. Logiko modula je mogoče razširiti glede na izbiro mikrokontrolerjev drugih proizvajalcev z večjim številom prioritetnih niti.

4.2.1 Tip in objekti

Osrednjo vlogo ima objekt s svojimi funkcionalnostmi. Vsak objekt pripada določenemu tipu. Tip definira lastnosti in funkcionalnosti, ki jih lahko izvajamo nad njegovimi instancami. Objekti, ki sodelujejo v sistemu prenašanja sporočil, morajo implementirati vmesnik naslovnika. Sem spadajo vsi objekti, katerih deskripcija tipa vsebuje metodo "MessageHandler" (vrstica 5 izseka kode 4.2, ki prikazuje deklaraciji tipa naslovnika in pa instance le-tega po imenu "Naslovnik_A").

Listing 4.2: Deklaracija tipa naslovnik in primer pripadajočega objekta.

```
1 typedef struct tipNaslovnik TipNaslovnik
2 typedef struct TipNaslovnik
3 {
4    TipNaslovnik *super;
5     Msg const *(*messageHandler)(void* self, Msg* message, char*
          returnValue);
```

```
} TipNaslovnik;

struct naslovnik_A {
    TipNaslovnik *tip; // kazalec na Tip deskriptor
    // ...
} Naslovnik_AA;
```

Ko objekt sprejme sporočilo, se nad njim pokliče metoda "MessageHandler" s sporočilom kot drugim parametrom. V tej metodi implementiramo funkcionalnosti glede na tip in vrednosti sporočila (kodni izsek 4.3). Posamezna naslovljena sporočila lahko smatramo kot zahteve za izvedbe funkcij nad objektom. V tem primeru tip sporočila predstavlja ime funkcije, medtem ko ostale vrednosti predstavljajo parametre klica.

Listing 4.3: Primer implementacije metode "MessageHandler" s switch stavkom v notaciji C programskega jezika.

```
Msg const *(*MSGHandler)(void* self, Msg* message, char*
    returnValue) {
    // ...
    switch(message->type) {
        case MSG_TYPE_0:
            // izvede se funkcionalnot ‘‘MSG_TYPE_0’’
        return 0;  // Funkcionalnost je v celoti izvedena.
        // ...
        case MSG_TYPE_N:
            // izvede se funkcionalnot ‘‘MSG_TYPE_N’’
        return message;  // funkcionalnost delno izvedene, kontrolo
            predajamo ‘‘messageHandler-ju’’
                        // v verigi tipov naslovnikov navzgor
                            (Naslovnik->super)
    }
}
```

Pri kreiranju novih tipov naslovnikov lahko dedujemo iz že obstoječih. V tem primeru nastavimo vrednost kazalca "super" (izsek kode 4.2, vrstica 4) v na novo kreiranem tipu na vrednost pozicije v pomnilniku, kjer se nahaja tip, iz katerega dedujemo. Z dedovanjem nov tip naslovnika pridobi množico funkcionalnosti. Določene funkcionalnosti lahko redefinira, pri čemer se lahko še vedno sklicuje na podedovane.

4.2.2 Sporočilo

Vsebino sporočila sestavljajo :

- kazalec na naslovljeni objekt,
- informacija o tipu,
- prioriteta,
- čas kreiranja (timeStamp),
- ter segment s poljubnimi podatki, ki jih pošiljamo naslovniku.

Na podlagi informacij o tipu sporočila se naslovnik odloči, katera akcija naj se izvede ob prejetju. Prioriteta sporočila služi kot informacija pri razvrščanju v prioritetni vrsti, v kateri sporočila čakajo na sprovedbo.

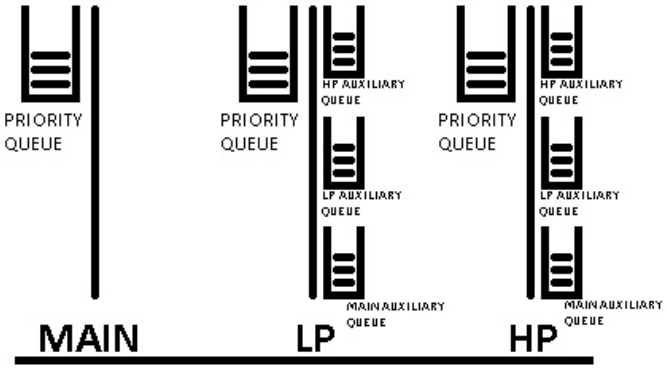
Procesni čas, porabljen za obdelavo prekinitvenega dogodka na nitih, kjer se zahteva visoka odzivnost, mora biti čim krajši. Kot odziv na prekinitveni dogodek lahko napravimo novo sporočilo, ki ga opremimo s časovnim markerjem in ostalimi informacijami ter ga pošljemo v nadaljnjo obdelavo, ki se izvede na niti z nižjo prioriteto. Na ta način sprostimo nit višje prioritete, ki s tem ostane v pripravljenosti na nadaljne dogodke.

4.2.3 Prekinitvene niti

V aktualni implementaciji obstajajo tri niti, na katerih lahko izvajamo aktivnosti. Nit z najnižjo prioriteto (glavna nit), ki se štarta na začetku aplikacije

in je aktivna vse do konca (ob ponovnem zagonu se zadeva ponovi). Medtem ko se višjeprioritetna oz. prekinitvena nit zažene glede na dogodek, ki predstavlja zunanjo oz. programsko prekinitveno zahtevo, in konča takoj, ko so vse zahteve obdelane. Potrebno je namreč prepustiti procesno moč nitim nižjih prioritet.

Kontrolo izvajanja funkcionalnosti nad objekti lahko prenašamo z niti na nit. Tako lahko s katerekoli niti podamo zahtevo za klic funkcije "FUNC_X" nad objektom "OBJ_X", ki naj se izvede na niti NIT_X. Zahtevo podamo z ukazom "postMessage(object, message, thread)".



Slika 4.3: Prioritetne in pomožne vrste, razporejene po nitih.

Vsaki niti pripada prioritetna vrsta, v katero se dodajajo sporočila prek klicev "postMessage" metode (slika 4.3). Sporočila se v vrsti sortirajo glede na informacijo o prioriteti. Z ukazom "getMessage" (izsek kode 4.4) iz vrste preberemo naslednje še neprebrano sporočilo. Funkcija vrne boolean vrednost, ki je vedno TRUE, razen v primeru, ko iz vrste preberemo sporočilo tipa "QUIT". Na niti z najnižjo prioriteto se klic izvede sinhrono in v primeru, da v vrsti ni nobenega sporočila, nit blokira. V tem primeru implementacija poskrbi, da mikrokontroler preide v spanje (SLEEP mode).

Vse to nam omogoča implementacijo programske zanke (izsek kode 4.4).

Listing 4.4: Zanka sporočilnega sistema.

```
1 // ...
2 while(getMessage(&message))
```

```
3     dispatchMessage(message.object, &message, 0);
4 // ...
```

Ko pošiljamo sporočilo z niti nižje na nit z višjo prioriteto, sporočilo zavedemo v prioritetno vrsto, ki pripada naslovljeni niti in sprožimo programsko prekinitveno zahtevo določene prioritete. Posledično se prekinitvena rutina zažene. Na prvem mestu rutine obstaja zanka (izsek kode 4.4), ki razpošlje vsa sporočila po posameznih objektih. Ko so vsa sporočila obdelana, se zanka prekine in nadaljuje se izvajanje prekinitvene rutine do konca.

Za razpošiljanje sporočil, potem ko jih preberemo iz prioritetne vrste, se pokliče metoda "dispatchMessage".

Ko iz prioritetne vrste preberemo sporočilo, pokličemo metodo "dispatchHandler" za razpošiljanje sporočil. Metoda se v zanki sprehodi po tip deskriptorjih hierarhije dedovanja naslovnika in izvaja metode "messageHandler", vse dokler ne obiščemo vseh nivojev oz. metoda ne vrne vrednosti 0, kar pomeni, da je sporočilo uspešno sprovedeno.

Listing 4.5: Razpošiljanje sporočil naslovljenim objektom in izvajanje akcij glede na njihov tip upoštevaje mehanizem dedovanja.

```
1  Msg* dispatchMessage(void* _self, Msg* message, char* returnValue)
2  {
3     Class *classPtr = *(Class **)_self;
4      // ...
5     for( ;classPtr ; classPtr = classPtr->super)
6     {
7        message = (Msg*)classPtr->messageHandler(_self, message,
             returnValue);
8        if(message == 0)
9           break;
10    }
11     // ...
12    return message;
13 }
```

Funkciji "postMessage" in "getMessage" dostopata do prioritetnih vrst z različnih niti. Zato moramo zaščititi te vrste s sinhronizacijskimi metodami. Ena izmed njih je "Lamport's bakery" algoritem. Pri zaščiti sekcij ob uporabi prioritetnih niti imamo naslednjo posebnost. V primeru, da nit nižje prioritete zaseže resurs in da se hkrati v tem trenutku zaradi poljubnega dogodka štarta nit višje prioritete, katera poskuša zavzeti ta isti resurs, vendar pri temu ne uspe. Če bi bil klic poskusa zasega sinhron, kar pomeni, da bi nit blokirala, dokler se ne sprosti resurs, na katerega nit, čaka bi prišlo do mrtvega objema, kajti nit z nižjo prioriteto ne more nadaljevati procesiranja, dokler obstatja katerakoli nit z višjo prioriteto aktivna.

Zaradi tega lahko niti višjih prioritet samo poskusijo rezervirati resurs in če zaseg ni uspešen, le nadaljujejo izvajanje po alternativni poti. V primeru, da "postMessage" ne uspe zaseči prioritetne vrste, vstavi sporočilo v pomožno vrsto, do katere dostopa le-ta nit (sekcije ni potrebno zaščititi). Vsaka, razen glavne niti, ima za ta namen eno pomožno vrsto za vsako od preostalih niti (slika 4.3).

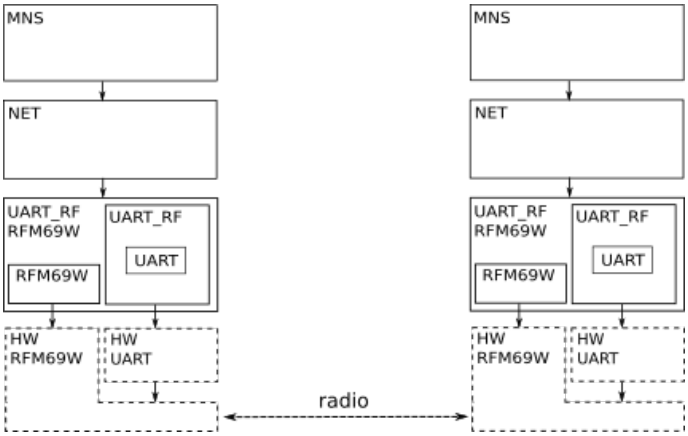
V trenutku, ko se prioritetna vrsta sprosti, izvedemo akcijo, katera prepiše sporočilo iz pomožne vrste v prioritetno. Ta akcija se izvede na niti, kateri pripada pomožna vrsta. To storimo z ukazom "postMessage", s katerim pošljemo sporočilo tipa "REPUMP_AUX".

4.3 Komunikacijski sklad

Poglavje opisuje programske module, ki so potrebni, da lahko med prostorsko distribuiranimi enotami razpošiljamo sporočila. Moduli so urejeni v skladovnico po odvisnosti drug od drugega.

Najnižje je postavljen modul za asinhrono serijsko komuniciranje, ki komunicira direktno s strojno enoto mikrokontrolerja, po imenu Uart (3.4). Gre za osnovno implementacijo Uart gonilnika oz. Uart-specializirano za komuniciranje preko radijskih frekvenc (UartRF_rfm69w/UartRF_bim2).

Osnovna implementacija Uart se uporablja pri komunikaciji preko Blue-



Slika 4.4: Komunikacijski sklad.

tooth modula. Medtem ko se UartRF_rfm69w implementacija uporablja za komunikacijo preko radijskega oddajno/sprejemnega modula rfm69w.

Nad modulom Uart je paketni modul po imenu Net, ki skrbi za prejemanje in pošiljanje paketov. Sledi modul za pošiljanje in sprejemanje sporočil Mns (MessageNetStream), katerega naloga je, da s pomočjo Net modula pošilja in prejema sporočila.

4.3.1 Modul za asinhrono serijsko komuniciranje (UART)

Uart objekt implementira programski vmesnik, po imenu ReceiverTransmitter. Moduli, ki uporabljajo Uart, dostopajo do njegovih funkcionalnosti preko tega vmesnika. V komunikacijskem skladu je tak modul Net.

ReceiverTransmitter objekt vsebuje dve opravilni metodi, ki morata modul, ki objekt uporablja, nastaviti. To sta dataReceiverHandler ter transmitDataHandler, preko katerih ReceiverTransmitter obvešča svojega uporabnika o prejetju podatkov ter zahtevi po novih podatkih za pošiljanje.

Poleg tega implementira metodi writeData ter readData za pisanje oz. branje iz vmesnih pomnilnikov. To so asinhrone metode, ki ne blokirajo izvajanja niti. Če npr. v primeru branja iz vmesnega pomnilnika ni podatkov, metoda enostavno vrne vrednost 0 prebranih bajtov.

IsAllDataSent metoda nam pove, ali je ReceiverTransmitter končal s

pošiljanjem. Metoda odgovori pritrdilno v primeru, da so programski in strojni vmesni spomin ter pomikalni register prazni.

Dodatno je na voljo skupek funkcij za kontrolo sprejemanja in oddajanja podatkov (startReceiving ...).

Tip, ali gre za radio komunikacijo oz. ne, je razviden iz radioExtension kazalca (vrstica 16 izseka 4.6).

Listing 4.6: Programski vmesnik za asinhrono pošiljanje podatkov, ki ga implementira modul Uart.

```
1  typedef struct __receiverTransmitter {
2     Class *pClass;
3     // void (*dataReceiveTimeout)(ReceiverTransmitter*, void* owner);
4     void (*dataReceivedHandler)(ReceiverTransmitter*, void* owner);
5     BOOL (*transmitEventEnable)(ReceiverTransmitter*, BOOL enable);
6     void (*transmitEventDisableCount)(ReceiverTransmitter*, BOOL
          incNotDec);
7     void (*transmitDataHandler)(ReceiverTransmitter*, void* owner);
8     char (*isAllDataSent)(ReceiverTransmitter*);
9     int (*writeData)(ReceiverTransmitter*, void* outputBuffer, int
          numberOfBytesToWrite);
10    int (*readData)(ReceiverTransmitter*, void* inputBuffer, int
          numberOfBytesToRead);
11    ReceiverTransmitterFunction startReceiving;
12    ReceiverTransmitterFunction stopReceiving;
13    ReceiverTransmitterFunction startTransmitting;
14    ReceiverTransmitterFunction stopTransmitting;
15    void* receiverTransmitterOwner;
16    RadioReceiverTransmitter *radioExtension;
17 } ReceiverTransmitter;
```

Implementacija modula Uart, ki je tudi osnova za nadaljnje dedovanje (Uart_RF), nudi naslednje konfiguracijske parametre:

- hitrost prenosa v baudih,

- dodaten vmesni pomnilnik za sprejemanje podatkov,
- sprejemna strojna prekinitvena prioritetna nit (strojni Uart modul),
- sprejemna programska prekinitvena prioritetna nit,
- oddajna programska prekinitvena prioritetna nit (strojna nit je privzeta).

Pri višjih baudnih hitrostih je časa med sprejemom dveh bajtov manj kot sicer. V tem času mora programska koda sprocesirati sprejet podatek iz RCREG-registra pred prihodom novega, in sicer v krajšem času, drugače pride do prekoračitve. Če je čas, potreben za procesiranje posameznega podatka, večji od tega časa oz. če hočemo procesno razbremeniti prioritetno nit in ob tem povečati odzivnost, potem je potrebno uvesti vmesni pomnilnik za sprejemanje podatkov. Samo procesiranje prejetih podatkov pa prenesti na nižjo prioritetno nit.

4.3.1.1 Sprejemnik

Ob prejetju podatka se izvede strojna prekinitev na predkonfigurirani niti, ki nakaže, da v RCREG-registru čaka podatek.

V verziji brez vmesnega pomnilnika se na isti niti pokliče metodo dataReceivedHandler. Metoda readData prebere in vrne podatek iz RCREG-registra. Po drugi verziji strojna prekinitvena nit podatek iz RCREG-registra vpiše v vmesni spomin FIFO izvedbe (prvi noter, prvi ven) s funkcijo storeData in na predkonfigurirani programski niti pokliče metodo dataReceivedHandler.

Dostop do vmesnega spomina je zaščiten s sinhronizacijskimi elementi, kajti do njega se dostopa iz različnih prioritetnih niti. Dostop do sekcije se izvaja preko metod readData ter storeData.

4.3.1.2 Oddajnik

Izbira prekinitvene niti, na kateri izvajamo odzivno rutino, ki zagotovi strojnemu modulu podatke za oddajo, je v trenutni različiči statično določena. Gre za t. i. nižjo prioritetno nit. Možnosti uporabe vmesnega oddajnega pomnilnika ni. Lahko pa določimo nit, na kateri se izvede transmitDataHandler, ki pove, da je oddajniku zmanjkalo podatkov za pošiljanje.

Strojna prekinitev se izvaja, vse dokler ne zapolnimo TXREG-registra. Ker pa želimo zahtevo po podatkih za pošiljanje izvesti na nižji prioritetni niti, je potrebno prekinitev na trenutni niti onemogočiti za čas izvedbe.

To storimo s klicem metode UART_transmitEventDisableCount(..., TRUE) pred preskokom na nižjo nit. Klic poveča spremenljivko disableEventCount za vrednost 1 in ko ta preseže 0, prekinitev onemogočimo. Takoj po izvedbi zahteve po podatkih na nižji prioritetni niti ponovno pokličemo metodo UART_transmitEventDisableCount(..., FALSE), tokrat s parametrom FALSE. S klicem se vrednost spremenljivke disableEventCount poveča in ko vrednost pride nazaj na 0, omogočimo prej onemogočeno prekinitev.

4.3.2 Modul za radijsko komuniciranje (Uart_RF)

Uart_RF-objekt implementira poleg programskega vmesnika po imenu ReceiverTransmitter še vmesnik RadioReceiverTransmitter (izsek 4.7). RadioReceiverTransmitter objekt je dostopen preko radioExtension kazalca ReceiverTransmitter objekta.

Uart_RF razširja funkcionalnost Uart modula za pošiljanje podatkov preko radijskega sprejemno/oddajnega modula. Vsakič ko iniciramo pošiljanje podatkov, se preko oddajnika pošlje predkonfigurirano število bajtov vrednosti 0x55. Ti podatki skupaj z metapodatki (start in stop bit) v podatkovnem oddajnem toku predstavljajo kontinuirano pravokotno nihanje. Ta signal je potreben, da se lahko bitni sinhronizator sprejemnega RF-modula sinhronizira z oddajnikom. Drugi pogoj za pravilno delovanje bitnega sinhronizatorja je, da se mora preostali tok podatkov po sinhronizaciji držati raznih stati-

stičnih okvirjev. Npr. povprečna vrednost koeficienta znak:presledek ne sme biti manjša od 30 % in ne večja od 70 % na določenem časovnem intervalu. V primeru, da moramo večkrat zapored poslati vrednost 0, temu pogoju ne bi zadostili.

Zato podatke pretvorimo v Manchesterjevo kodo, kjer je bit 1 predstavljen s sekvenco dveh bitov 0, 1 ter bit 0, predstavljen z 1, 0. S tem dosežemo vrednost koficienta znak:presledek enako 50 %. Vse to se zgodi za ceno podvojene dolžine podatkov.

Kot je že omenjeno, naša rešitev uporablja rfm69w modul v načinu delovanja, ki zagotavlja pošiljanje/sprejemanje podatkov v realnem času. Zato moramo manchester kodiranje ter pošiljanje uvodnih podatkov (preamble) opraviti programsko. Modul ima sicer te funkcije implementirane v strojni kodi v načinu s paketnim upravljanjem, ki ne omogoča komuniciranja v realnem času.

Listing 4.7: Programski vmesnik za radijsko asinhrono pošiljanje podatkov, ki ga implementira modul Uart_RF.

```
1  typedef enum __endListenAction {
2      RRT_LISTEN_ACTION_RESTART,
3      RRT_LISTEN_ACTION_RX_START,
4      RRT_LISTEN_ACTION_STOP
5  } EndListenAction;
6  typedef struct __listenConf {
7     ULONG rx_time; // us
8     ULONG idle_time;
9      EndListenAction action;
10     ULONG rssiThresh_dataTimeout;
11     EndListenHandler endListenHandler;
12 } ListenConf;
13
14 typedef void (*ListenFunc)(ReceiverTransmitter* rt, ListenConf*
       listenConf);
15 typedef void (*RssiThresh_dataReceived)(ReceiverTransmitter* rt,
```

```
    void *argument);
16 typedef struct __radioReceiverTransmitter
17 {
18   ListenConf listenConf;
19   ListenFunc startListen;
20   // resets RSSI timeout
21   RssiThresh_dataReceived rssiThresh_dataReceived;
22 } RadioReceiverTransmitter;
```

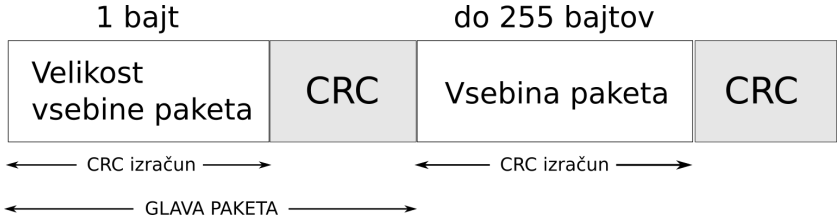
RadioReceiverTransmitter vmesnik omogoča uporabo intervalnega poslušanja — funkcionalnost, ki jo omogoča modul rfm69w (slika 3.5). V tem načinu je mogoče privarčevati veliko energije, in sicer v primeru, ko mora biti enota ves čas pozorna na prihajajoče podatke. Enota vklaplja ter izklaplja sprejemnik v določenih časovnih intervalih. Ko se sprejemnik vklopi, enota preverja indikator moči prejetega signala RSSI. V primeru, da indikator preseže mejno vrednost, enota ostane v načinu sprejemanje, v nasprotnem primeru po izteku časovne kontrole (vrstica 7 izseka 4.7) preide v neaktivno stanje, kjer ostane določen čas (vrstica 8 izseka 4.7), dokler se postopek ne ponovi.

Ko indikator moči preseže mejno vrednost, se vključi časovna kontrola (vrstica 10 izseka 4.7), ki ob izteku, v primeru, da paketa s podatki nismo prejeli, izvede akcijo EndListenAction (vrstica 5 izseka 4.7). S tem se izognemo temu, da enota ostane v načinu sprejem, kljub temu da nismo uspeli sprejti želenih podatkov.

Enako se zgodi ob prejetju paketa, kar programska koda uporabnika naznani s klicem funkcije vmesnika rssiThresh_dataReceived (vrstica 21 izseka 4.7). Le uporabniška programska koda ima vedenje o semantični pravilnosti sprejetega sklopa podatkov. Obstajajo tri možne akcije, in sicer nadaljuj v načinu sprejemanje, ponovno zaženi intervalno poslušanje, nato preide v neaktivno stanje.

4.3.3 Paketni protokol (Net)

Paketni modul ima sprejemno ter oddajno čakalno vrsto. Pri pošiljanju v oddajno čakalno vrsto vstavimo podatke, ki jih želimo poslati, nato iniciramo pošiljanje. Preden se podatki začnejo pošiljati, se tem dodajo še metapodatki. Najprej se izračuna 2 bajta dolga ciklična redundančna koda, ki se jo doda na koncu vrste. Nato se na začetku vrste doda še glava paketa, ki vsebuje dolžino podatkov. Glavi paketa prav tako izračunamo ter dodamo CRC-kodo (slika 4.5).



Slika 4.5: Vsebina paketa.

Pri sprejemanju začnemo z branjem glave paketa. Pri vsakem sprejetem bajtu se vsebina sprejemnne vrste pomakne za en bajt. Nato preverimo izračun CRC-kode za določeno število bajtov. Pri sprejemanju glave paketa je to število enako 3 (dolžina paketa + CRC-koda). V primeru, da izračun CRC-kode ne ustreza, je potrebno počakati na naslednji sprejet podatek. V nasprotnem primeru smo uspešno prejeli glavo paketa, ki narekuje dolžino preostalih podatkov v sklopu tega paketa. Nadalje računamo CRC-kodo proti omenjeni dolžini in če se izračun izide, smo prejeli paket. Če pa se izračun ne izide, se postopek začne znova z branjem glave paketa.

4.3.4 Modul za pošiljanje ter sprejemanje sporočil (MessageNetStream)

Modul ima sprejemni ter oddajni vmesni pomnilnik. Gre za vrsti tipa FIFO (prvi noter, prvi ven), ki sta namenjeni za hrambo sporočil. Poleg sporočila se v vrstah beleži tudi podatek o naslovniku oz. prejemniku. Ta je podan

s posebnim lokacijskim identom, ki predstavlja oddaljeno enoto. Vrednost identa 0 je rezervirana za razpršeno naslavljanje (broadcast) pri oddajanju sporočila. Sprejeto sporočilo je poleg z identom pošiljatelja, povezano tudi z časovnim markerjem, ki predstavlja čas trenutka sprejema. Modul podatke sporočil prenaša preko paketnega modula, ki se nahaja v skladu en nivo pod njim. Sporočila se pri oddaji pretvorijo v pakete ter pri spremjemu nazaj iz sprejetih paketov.

Funkcija odmev (ECHO) naslovljeni enoti pošlje sporočilo tipa zahteva za odmev ECHO_REQUEST. Ko naslovljena enota sporočilo prejme, takoj odgovori pošiljatelju s sporočilom odmev (ECHO_RESPONSE). S tem pošiljatelj preverja dosegljivost določene lokacijske enote (naslovnika). Poleg tega lahko izmeri čas sporočila na poti k naslovniku. Ta čas uporabljamo pri usklajevanju med uro centralne enote ter urami senzorjev.

Lastnik modula je objekt tipa TipNaslovnik (izsek 4.2), katerega modul obvešča o raznih dogodkih:

- sprejetje sporočila, normalnega, odmeva (ECHO_RESPONSE) ali zahteve za odmev (ECHO_REQUEST),
- oddajna čakalna vrsta je prazna, potrebno je vstaviti novo sporočilo,
- iztek časovne kontrole pri sprejemu sporočila.

4.4 Oštevilčenje enot in časovna sinhronizacija

Osrednji entiteti sistema za merjenje sta IR-gospodar ter IR-suženj. Skrbita za povezovanje centralne enote in senzorjev ter pošiljanje meritvenih dogodkov. Vsaka enota, bodisi centralna enota bodisi senzor ob zagonu, pridobi enolično številko, ki predstavlja njeno sejo skozi celoten proces meritev. Gre ali za število pseudo generatorja s semenom, ki predstavlja neko izmerjeno vrednost z visoko resolucijo na lokaciji, kjer je enota postavljena (npr. RSSI

identifikator) ali za zaporedno število, ki ga pridobimo na podlagi inkrementiranja prejšnjega identa, ki je zapisan v EEPROM-pomnilniku, v katerem preživi čas med dvema sejama.

Poleg tega ima vsaka enota svoj statični ident, določen ob njenem nastanku v proizvodnji liniji (npr. strojni ident komunikacijskega modula) oz. določen v programski kodi.

Vsaka enota se pri komunikaciji vedno predstavlja s statičnim identom. Poleg tega se preverja tudi seja enote. Npr. med zagoni ima centralna enota različne idente sej in pri sprejemanju dogodkov prekinitev se to upošteva. Tako se dogodki prekinitev, ki so naslovljeni na določeno centralno enoto različne seje, kot je bila prevzeta ob oštevilčenju, zavržejo. Ponastavitev centralne enote v tem primeru pomeni tudi ponastavitev senzorja. Senzor je potrebno ponovno oštevilčiti (lista senzorjev na centralni enoti se nahaja v statičnem pomnilniku in se ob ponastatitvi zavrže).

Za medsebojno komuniciranje, IR-gospodar in IR-suženj implementirata protokol za izogibanje trkov. Akterji v komunukaciji se morajo med seboj nekako sporazumeti, kdo lahko oddaja podatkovne pakete v določenem trenutku. Sicer prihaja do komunikacijskih trkov, iz katerih se ne da razbrati informacij. Predvideti je potrebno delovanje več enakih instanc sistema na istem področju. Sistemi se morajo med seboj sporazumevati, da bi zmanjšali število komunikacijskih trkov. V te namene se uporabljajo t.i. komunikacijski okvirji (obrazložitev v naslednjih odstavkih).

Preden lahko z meritvenim sistemom opravimo časovno meritev, se mora centralna enota povezati s senzorji. Omenjena funkcionalnost se imenuje oštevilčenje senzorjev. Težava pri tem postopku je ta, da ni mogoče za vsako meritev zadostiti pogoju, kjer so vsi senzorji v trenutnem radijskem dometu glede na lokacijo centralne enote. Zato mora biti postopek inkrementalen. Oštevilčenje namreč poteka v večih iteracijah glede na pozicijo postavljenih senzorskih enot. IR-gospodar vodi evidenco o do sedaj opravljenih iteracijah oštevilčenja senzorjev. V okviru inicirane poizvedbe pošlje z razpršenim pošiljanjem vsem enotam v radijskem dosegu

sporočilo "poizvedba po neoštevilčenih enotah". Sporočilo vsebuje zgoraj omenjen ident gospodarja, ident oštevilčenja, inkrementalni ident podpoizvedbe oštevilčenja ter število okvirjev za odgovor.

Na to sporočilo odgovorijo vse enote, ki še niso bile oštevilčene v okviru trenutnega oštevilčenja. Vsaka izmed teh enot si z z uporabo pseudo generatorja izbere naključno število, ki predstavlja N-ti okvir za odgovor.

Nato čaka na poizvedbo s strani centralne enote (gospodarja), ki z dodatnim sporočilom vpraša, komu je bila dodeljena določena številka okvirja i. Centralna enota z razpršenim pošiljanjem (naslovljene so vse enote oz. naslovnik paketa ni določen) pošlje poizvedbo za vsak izmed N določenih okvirjev, namenjenih za odgovor enot. Kljub temu v gospodarjevih poizvedbah za določene okvirje prihaja do trkov. Postopek je potrebno ponavljati, vse dokler niso vse enote v radijskem dometu uspešno oštevilčene.

Nato je potrebno uskladiti ure oštevilčenih senzorjev z uro centralne enote. To storimo tako, da izmerimo čas, ki je potreben za pošiljanje paketa z informacijo o dveh časovnih objektih od centralne enote do senzorjev. Imamo dve možnosti in sicer, da vrednost določimo pri izdelavi programske opreme oz. da dodamo funkcionalnost, ki na podlagi pošiljanja dejanskih paketov omogoča izračun te vrednosti, tik preden se izvede časovna uskladitev.

Funkcionalnost se izvede s poizvedbo lokalne ure v trenutku pošiljanja paketa s sporočilom tipa "prošnja za odziv" (echo request) ciljni enoti, ki se na to odzove z odpošiljanjem sporočila "odziv". V trenutku sprejema sporočila ponovno napravimo poizvedbo lokalne ure ter ti vrednosti odštejemo prej poizvedeno vrednost lokalne ure. Razlika predstavlja čas pošiljanja in sprejemanja omenjenih paketov. Razliko pomnožimo z vrednostjo, ki predstavlja razmerje med časom pošiljanja ter časom potovanja paketa v obe smeri. Ta vrednost je odvisna od programske kode. V večini primerov jo lahko z določeno napako zaokrožimo na 50 %.

Slednja metoda omogoča natančno časovno usklajevanje v primeru programsko oz. strojno heterogenih tipov senzorjev. V strojno oz. programsko heterogenem sistemu mora gospodar za vsak izmed oštevilčenih senzorjev

izmeriti čas pošiljanja. Ko je vrednost izmerjena IR-gospodar napravi poizvedbo svoje lokalne ure ter sestavi podatkovni paket, ki vsebuje to vrednost, vključno z izmerjeno vrednostjo trajanja pošiljanja paketa senzorju.

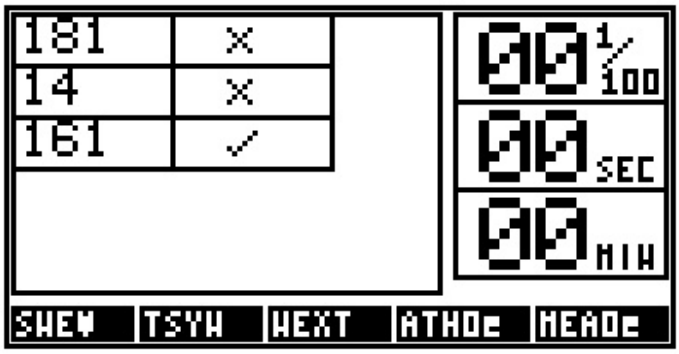
Sledi pošiljanje paketa časovne uskladitve senzorski enoti, ki ta paket prejme ter ponastavi svojo uro na vrednost lokalne ure centralne enote v trenutku začetka pošiljanja plus vrednosti časa trajanja pošiljanja paketa. Pri tem postopku gre gospodar skozi listo vseh oštevilčenih enot ter vsaki izmed njih pošlje zahtevo za uskladitev. Tudi proces usklajevanja ur poteka inkrementalno. Časovna usklajevanja imajo svoj ident. Suženj ob sprejemu zahteve za uskladitev preveri, ali je že časovno usklajen glede na par vrednosti (seja gospodarja, ident časovne uskladitve). Gospodar inicira nov val časovnih uskladitev z inkrementiranjem omenjenega identa.

Gospodar vodi evidenco o senzorskih enotah, s katerimi je v trenutni seji povezan (preko oštevilčenja) v posebni listi. Vsaka enota v listi je uparjena z lokalno uro v trenutku časovne uskladitve ter časom, po preteku katerega je absolutna napaka ure senzorja izven dupustnega intervala (zopet uporabno v heterogenem sistemu senzorjev). Tako lahko program obvešča uporabnika o zahtevanih časovnih uskladitvah oz. te uskladitve opravi z uporabo časovnikov kar sam.

Uporabniški vmesnik za oštevilčenje ter časovno usklajevanje s prikazom liste do tega trenutka oštevilčenih enot na sliki 4.6. Vsaka enota v listi je predstavljena z identom ter zastavico, ki prikazuje status časovne uskladitve. V kolikor enota ni usklajena oz. je od zadnje uskladitve poteklo preveč časa, je omenjeni status enak nič (križec).

Suženj se v določenem trenutku lahko nahaja v enem izmed treh stanj, in sicer so to stanje nepovezan, povezan in časovno neusklajen ter časovno usklajen. Kot prekinitvena enota, ki vzorči podatke senzorjev (IR-vrat oz. tračnega stikala), se suženj aktivira v stanju "časovno usklajen".

Suženj ima listo dosedaj gospodarju še neodposlanih prekinitvenih dogodkov. Dogodek v listi je predstavljen z vrednostjo lokalne ure, katere poizvedba se napravi v trenutku začetka odziva prekinitvene rutine na pre-



Slika 4.6: Oštevilčenje in časovno usklajevanje sužnjev.

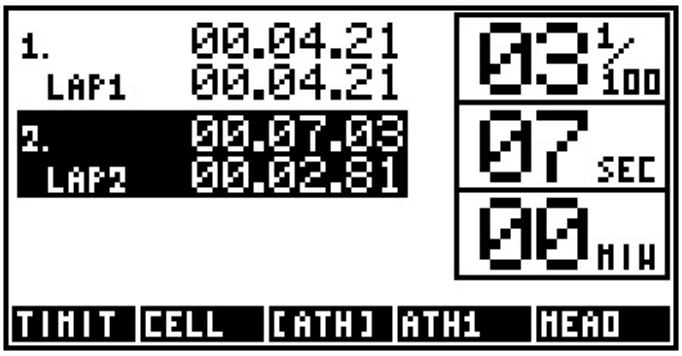
kinitveni dogodek meritve. Vse dokler vrsta ni izpraznjena, suženj ponavlja sledeči postopek:

- gospodarju pošlje sporočilo na podlagi informacije o prvem dogodku iz liste,
- preide v način sprejemanja v katerem pričakuje potrditev gospodarjevega sprejema poslanega sporočila,
- v kolikor potrditev sprejme, nadaljuje z naslednjim dogodkom iz liste in če je lista izpraznjena, prekine sekvenco pošiljanja dogodkov,
- v kolikor v določenem času potrditve ne sprejme, zgenerira naključno število, ki predstavlja N-ti časovni okvir od tega trenutka dalje in preide v spanje, dokler ne poteče ta čas ter nato postopek ponovi.

V trenutku sprejema prekinitvenega dogodka gospodar le-tega aplicira v modul kronometer, kjer se sortiran vnese v shrambo časovnih dogodkov trenutne bodisi že zaključene meritve. Slednje se zgodi, v kolikor ga ni bilo mogoče takoj dostaviti.

V sliki 4.7 je prikazan uporabniški vmesnik, ki predstavlja funkcionalnosti za uporabo kronometra ter za pregledovanje meritvenih podatkov.

V načinu kronometer imamo tri osnovne ukaze, in sicer START za pričetek, LAP kot vmesni čas ter STOP za konec meritve.



Slika 4.7: Uporabniški vmesnik.

4.5 Povezovanje z zunanjimi napravami

Zunanje naprave se lahko na sistem za merjenje priključijo preko bluetooth naprave. Na uart vmesniku bluetooth modula se vzpostavi RFCOMM-emulacija RS232 serijskega porta. Mikrokontroler z bluetooth modulom komunicira preko strojnega HW_Uart modula. Komunikacijski sklad sestoji iz Uart objekta, RN42 objekta, ter razpošiljatelja ukazov (cmdDispatcher). Objekt Uart je gonilnik, ki upravlja z mikrokontrolerjevim HW_Uart modulom.

Naslednji v skladu je objekt RN42, ki upravlja z bluetooth modulom. Objekt izvaja ukaze v bluetooth ukaznem načinu ter vzorči dogodke, kot so vzpostavitev oz. prekinitev povezave, ponovni zagon itn. Ob inicializaciji modula preko ukazov nastavimo ime BT-modula, omogočimo pošiljanje zgoraj omenjenih dogodkov, po potrebi postavimo modul v način za varčevanje z energijo ter mnogo več. Po končani inicializaciji se modul postavi v podatkovni način. Tako modul deluje kot podatkovni kanal. Ko modul sprejema podatke, odstrani BT-metapodatke ter preusmeri uporabniške podatke na Uart vmesnik. Ko podatke pišemo preko Uart vmesnika, modul sestavlja BT-pakete ter jih pošilja preko BT-brezžične povezave.

Podatke, ki prihajajo iz RN42 podatkovnega objekta, nadalje obdeluje naslednji modul v skladu, t. i. razpošiljatelj ukazov. Razpošiljatelj s pomočjo ukaznega razčlenjevalca (parser) iz uporabniških podatkov izlušči ukaze.

Imamo tri tipe nizov, proti katerim razčlenjujemo (parse) sprejete podatke:

- "LIST()",
- "LIST(%(objekt)s)",
- "C(%(objekt)s, %(metoda)s, %(atributi)s, %(nit)s)".

Če se razčlenjeni niz ujema z nizom tipa LIST(), potem se razpošiljatelj ukazov odzove z listo vseh objavljenih objektov [obj0,obj1,obj2 ...]. Zunanja naprava lahko z ukazom LIST(%(objekt)s) nadalje pregleda listo vseh objavljenih metod določenega objavljenega objekta. Rezultat je lista metod v obliki [meth0,meth1,meth2 ...].

Pri ujemanju uporabniških podatkov s tipom niza C(%(objekt)s, %(metoda)s,-%(atributi)s, %(nit)s) je razčlenjevalec ukazov izluščil naslednje objekte: objekt, metoda atribut ter nit. Te objekte uporabi za klic funkcije preko Sistema prenašanja sporočil 4.2. Klic izvede z ukazom "postMessage(objekt, message, thread)".

Vračanje rezultatov zgoraj omenjenih ukazov razpošiljatelj ukazov oz. naslovljeni objekt izvede z ustavljanjem bodisi znakovnih nizov bodisi t. i. generatorjev nizov v oddajni vmesni pomnilnik.

Zaradi omejene kapacitete statičnega pomnilnika je nemogoče v oddajni vmesni pomnilnik vstaviti nek zelo dolg podatkovni niz (npr. listo meritvenih prekinitvenih dogodkov določenega atleta). Zato moramo v tem primeru uporabiti generatorje nizov. Velikost generatorjev je omejena, njegova logika pa je sposobna preletavati liste ter z uporabo minimalnega vmesnega pomnilnika formirati daljše podatkovne nize. Modul implementira generične generatorje, katerih sposobnost je predstavljanje heterogenih list.

Centralna enota objavlja sledeče objekte:

- kronometer z metodami start, lap, stop,
- shramba meritvenih dogodkov z metodami "athletUp", "athletDown",

Slika 4.8: Uporabniški vmesnik na zunanji enoti (ANDROID).

"measurementUp", "measurementDown", "timeEventUp", "timeEvent-Down",

- iterator atletov, meritev ter meritvenih dogodkov z možnostjo registriranja na spremembe,
- IR-gospodar z metodami "startEnumerate", "startTimeSync",
- lista sužnjev z metodo "clear" ter možnostjo registriranja na spremembe,

Poglavje 5

Testiranje

Kot zadnje dejanje nam preostane preizkus prototipnih enot v realnem okolju. Testiranje opravimo s centralno enoto CE ter dvema senzorjema S0, S1. Izmerimo naslednje testne elemente:

- radijski doseg,
- napake pri časovnih uskladitvah ur med centralno enoto ter senzorji,
- odstopanja med urami časovno usklajenih enot po preteku določenega časa,

5.1 Merjenje radijskega dosega enot

Z naslednjo meritvijo poskušamo prikazati vpliv razdalje na kakovost komunikacije med enotami. V testu uporabimo dve enoti, in sicer enoto A, ki v načinu oddaje pošilja podatkovne pakete, ter enoto B, ki te pakete sprejema.

Enota A naenkrat pošlje enoti B 30 paketov. Test izvedemo ob naslednjih pogojih:

- uporabimo najvišjo moč oddajnika, ki meri 13 dBm,
- dolžina uvodne besede (preamble) je 50 bajtov,

- prag indikatorja moči (RSSI) na sprejemniku je nastavljen na najnižjo vrednost, katere statične motnje (šum) ne presežejo (v našem primeru -96),
- hitrost komunikacije je 9600 baudov.

Test ponovimo na različnih razdaljah med enotama. Naslednja tabela 5.1 prikazuje rezultat, kjer je za vsako razdaljo d prikazana vrednost moči signala (RSSI) na sprejemniku, kakor tudi število uspešno prejetih paketov N.

d (m)	0	10	20	30	40	50	60	70	80	90	100
RSSI (dB)	-25	-49	-60	-64	-68	-75	-73	-76	-78	-78	-78
N	30	30	30	30	30	30	30	30	30	30	30

d (m)	110	120	130	140	150	200	290	300	320	340
RSSI (dB)	-85	-82	-80	-85	-83	-91	-92	-93	-92	-92
N	30	30	30	30	30	30	30	28	23	17

d (m)	350	380	400
RSSI (dB)	-93	-95	-96
N	14	8	1

Tabela 5.1: Statistika uspešnosti sprejemanja paketov na enoti B.

Največja izmerjena razdalja, pri kateri lahko med enotami še vzpostavimo komunikacijo, je 400 metrov. Izkaže se, da je ponekod signal na daljši razdalji med enotama močnejši od signala na krajši.

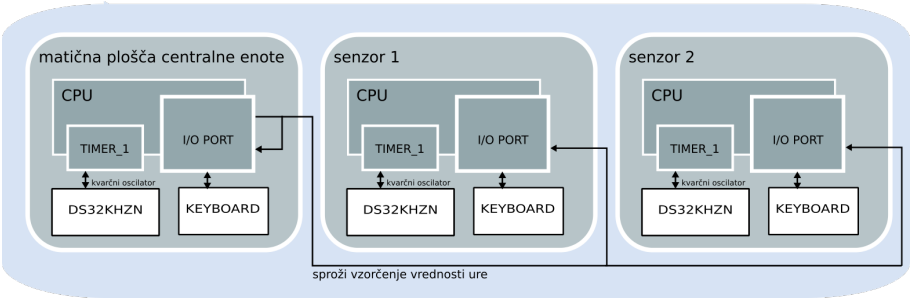
5.2 Merjenje napak pri časovnih uskladitvah

Namen tega testa je preveriti algoritem za časovno uskladitev ur senzorjev z uro centralne enote. Pred izvedbo testa s strani centralne enote oštevilčimo senzorje ter sprožimo proces časovne uskladitve med uro centralne enote ter urama senzorjev.

Nato z dodatno tipko, ki je nameščena na matični plošči centralne enote in povezuje vse enote (senzorje), iniciramo zunanji dogodek, ki je časovno

enak za vse. V trenutku, ko se omenjeni dogodek zgodi, vsaka izmed enot vzorči trenutno vrednost lokalne ure.

Mikrokontrolerji posameznih enot ta dogodek zaznajo kot prekinitev na enih izmed prekinitvenih vrat (slika 5.1). Prekinitvena zahteva je obdelana na niti z najvišjo prioriteto. Tu se s pomočjo metode "getTimeStamp" prebere trenutna vrednost lokalne ure ter se na glavno nit pošlje zahtevo za grafični prikaz te vrednosti.



Slika 5.1: Strojna implementacija sprožilca za vzorčenje vrednosti ur na povezanih enotah.

Ko imajo enote lastne izvore urinega takta, moramo upoštevati napako zaradi zamika med takti enot. Ta razlika vpliva na rezultat same časovne uskladitve, kakor tudi na samo preverjanje razlik med usklajenimi urami.

Omenjena napaka leži na odprtem intervalu $[0, t_{CLK_0})$, kjer je $t_{CLK_0} = 0.0305ms$.

Dodatna napaka se skriva v izmeri časovne dolžine ECHO-procesa, ki jo uporabimo kot osnovo za izračun vrednosti časa trajanja pošiljanja SYNC-sporočila za uskladitev ur med enotami (slika 5.2).

V kolikor omenjena vrednost ni deljiva z velikostjo urine periode t_{CLK_0} prihaja do napak zaradi diskretizacije.

Za čas trajanja pošiljanja SYNC-sporočila (T_{SYNC}) vzamemo vrednost $T_{ECHO}/2$. Tu imamo dodatno napako, in sicer zaradi zaokroževanja navzdol pri deljenju z 2.

Napaka pri izmeri T_{ECHO} je za njene različne vrednosti navzdol omejena z vrednostjo $-t_{CLK_0}$ ter navzgor z vrednostjo t_{CLK_0}, medtem ko je napaka

pri izračunu vrednosti $T_{ECHO}/2$ navzdol omejena z vrednostjo $-t_{CLK_0}$ ter navzgor z vrednostjo $0.5 \cdot t_{CLK_0}$.

Zamik začetka procesa uskladitve ur $SYNC_{offset}$ (slika 5.2) znotraj urine periode naknadno dodaja napako $(-t_{CLK_0}, 0]$. Proces uskladitve se začne z vzorčenjem lokalnega časa s funkcijo getTimeStamp(), katera vrne enako vrednost ure povsod znotraj iste urine periode ne glede na zamik (napaka diskretizacije).

Tako je skupna napaka upoštevaje napako pri izmeri dolžine SYNC-procesa, napako zamika začetka SYNC-procesa znotraj urine periode in zamika med takti ur navzdol omejena z vrednostjo $-2 \cdot t_{CLK_0}$ ter navzgor z $1.5 \cdot t_{CLK_0}$.

Torej vrednosti razlik, pridobljenih z vzorčenjem na osnovi referenčnega zunanjega dogodka, ležijo na diskretnem intervalu [-2,2] (slika 5.2).

Od tu sledi, da je nemogoče pričakovati do številke natančen rezultat uskladitve lokalnih ur.

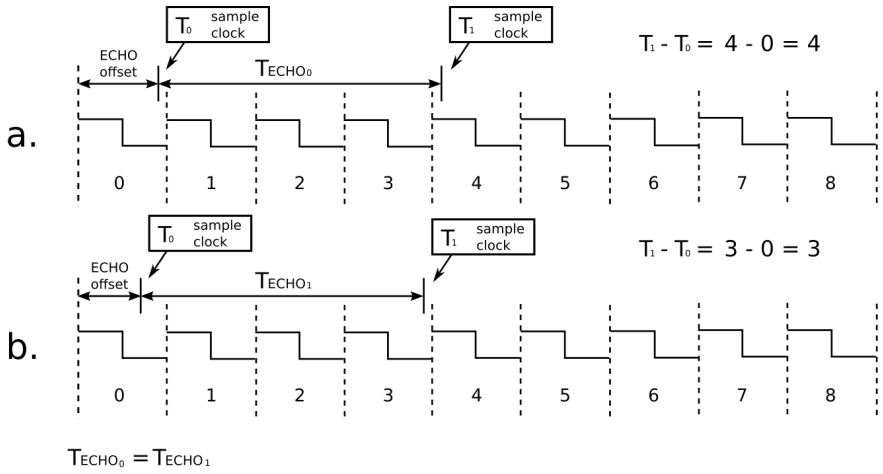
V okviru merjenja napak časovnih uskladitev napravimo štiri meritve. V vsaki izmed meritev napravimo 30 časovnih uskladitev med centralno enoto in senzorji ter odčitamo razliko med vrednostmi lokalnih ur centralne enote ter določenega senzorja.

$\lvert TM_{[T_{refce}]} - TM_{[T_{refs_0}]}\rvert$	0	1	2
$N[sync_{s_0}]$ (ms)	$[0, 0.0305)$	$(0, 0.0610)$	$(0.0305, 0.0915)$

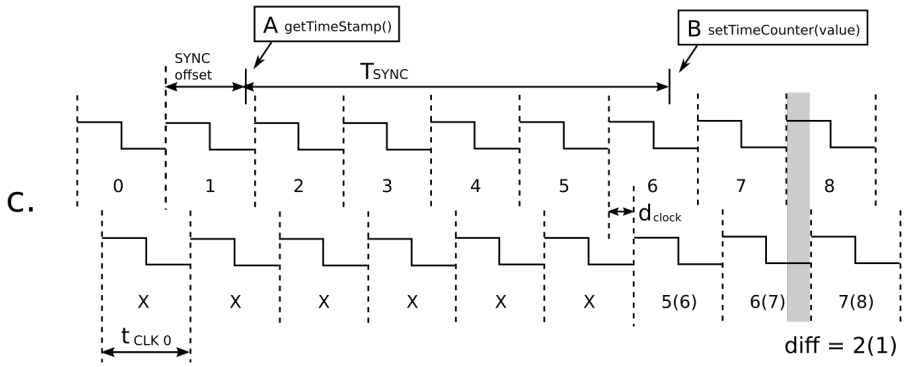
$\lvert TM_{T_{refce}} - TM_{T_{refs_0}}\rvert$	3	4
$N[sync_{s_0}]$ (ms)	$(0.0610, 0.1220)$	$(0.0915, 0.1525)$

Tabela 5.2: Velikost napake pri časovni uskladitvi glede na z zunanjim referenčnim dogodkom vzorčeno razliko med uro centralne enote ter uro senzorja.

Prva meritev je pri normalnih pogojih, pri katerih indikator moči signala, ki ga povzročajo statične motnje meri okoli -67 dB, ter aktivnega signala z vsebino okoli -15 dB. Enote uporabljajo lastne izvore (oscilatorje) za urin takt. Vrednosti napak se gibljejo na diskretnem intervalu [-1, 2] s porazdelitvijo, ki je prikazana v tabeli 5.3. Velikosti napak v ms glede na vzorčeno



sliki a. in b. prikazujeta različni vrednosti izmere časa pošiljanja/sprejemanja ECHO sporočila; vrednost zavisi od dolžine odmika štarta ECHO procesa od začetka urine periode



slika c. prikazuje situacijo, v kateri se ure enot na določenem intervalu razlikujejo za 2 urini periodi; zahtevan pogoj:

$$T_{SYNC} - \text{floor}(T_{SYNC} / t_{CLK\,0}) < t_{CLK\,0} \text{ AND } T_{SYNC} - \text{floor}(T_{SYNC} / t_{CLK\,0}) - (t_{CLK\,0} - SYNC_{offset}) > d_{clock}$$

$ECHO_{offset}$ - zamik štarta procesa ECHO_CALC znotraj urinega cikla

T_{ECHO} - dejanska dolžina ECHO_CALC procesa

- območje z najvišjo urino razliko

d_{clock} - zamik med takti urinih izvorov

$SYNC_{offset}$ - zamik štarta procesa SYNC_TIME znotraj urinega cikla

T_{SYNC} - dejanska dolžina SYNC procesa

Slika 5.2: Napake pri uskladitvi časa enot z lastnim izvorom urinega takta.

razliko med urami so podane v tabeli 5.2.

Pri drugi meritvi simuliramo okolje, kjer omejimo moč signala na sprejemniku. To dosežemo z dušenjem signala z uporabo kartonske škatle, ovite v aluminijasto folijo. Vanjo postavimo eno izmed enot. Moč signala, ki prodre

$N[sync_{s_0}]$	-1	0	1	2
n	2	18	5	5

Tabela 5.3: Porazdelitev napak pri časovni uskladitvi ob normalnih pogojih.

skozi škatlo, je občutno manjša od signala zunaj nje. Merjen RSSI-indikator znotraj škatle dosega vrednosti od -55 do -45 dB, medtem ko je vrednost zunaj nje okoli -16 dB. Vrednosti merjenih napak se prav tako gibljejo na intervalu [-1, 2]. Porazdelitev je prikazana v tabeli 5.4. Velikosti napak v ms glede na vzorčeno razliko med urami so podane v tabeli 5.2.

$N[sync_{s_0}]$	-1	0	1	2
n	2	12	13	2

Tabela 5.4: Porazdelitev napak pri časovni uskladitvi, kjer s škatlo, ovito v aluminijasto folijo, simuliramo okolje z veliko motnjami signala.

Med meritvama ni videti bistvenih razlik, ki bi potrjevale vpliv motenj na točnost časovnih uskladitev, še posebej, če upoštevamo napake zaradi zamika med takti ur enot.

Naslednja meritev je napravljena v normalnih pogojih, kjer vse enote uporabljajo isti izvor urinega takta. Tu je napaka zaradi zamika urinih taktov izključena. Vrednosti napak se v tem primeru gibljejo na diskretnem intervalu [-1, 2]. Porazdelitev je prikazana v tabeli 5.5. Velikosti napak v ms glede na vzorčeno razliko med urami so podane v tabeli 5.2.

$N[sync_{s_0}]$	-1	0	1	2
n	1	27	0	2

Tabela 5.5: Porazdelitev napak pri časovni uskladitvi z uporabo istega izvora urinega takta.

Rezultat zadnje meritve prikazuje bistveno boljši rezultat od prejšnih dveh meritev. Glede na to ocenjujem, da so časovne uskladitve zelo natančne. Razlog za majhna odstopanja najverjetneje leži v časovno nedeterministični programski kodi.

Kot zadnja nastopi meritev, pri kateri nastopata dva senzorja. Vsi skupaj

s centralno enoto se napajajo z istim urinim taktom. Cilj meritve je določiti izvor napak, ki prihajajo bodisi iz s centralne enote bodisi posameznih senzorjev. Porazdelitev je prikazana v tabeli 5.6. Iz rezultata lahko povzamemo,

$N[sync_s]$	-1	0	1	2	3	4
n (s_0)	2	22	1	2	2	1
n (s_1)	1	23	1	1	3	1

Tabela 5.6: Porazdelitev napak pri časovni uskladitvi centralne enote z dvema senzorjema (isti izvor urinega takta.)

da je v največji meri za napake kriv časovni nedeterminizem programske kode centralne enote. Sicer napake izvirajo tudi s strani senzorjev.

5.3 Odstopanja med urami časovno usklajenih enot

Vsaka enota ima svoj izvor urinega takta. Zato uporablja oscilator "DS32KHZ" (podpoglavje 3.2), ki deluje z relativno napako 2 ppm. Absolutna napaka lokalne ure tako narašča s časom, ki ga merimo od dogodka časovne uskladitve. Za najboljšo oceno te napake bi potrebovali točen referenčni čas. Tega v našem primeru testiranja nimamo, zato lahko ocenimo spodnjo vrednost napake. To storimo tako, da v določeni točki vzorčimo vrednosti ur dveh enot ter izračunamo razliko med njima. Spodnja meja absolutne napake v trenutku vzorčenja znaša 1/2 omenjene razlike. Prvi test izvedemo tako, da znaša temperatura ob posameznih urinih izvorih 22.5 ℃. Odstopanja na različnih časovnih točkah, merjenih od trenutka časovne uskladitve, so prikazana v tabeli 5.7. Izmerjeni časi senzorskih enot so prikazani kot razlika med uro senzorja in uro centralne enote.

Drugi test opravimo tako, da enega izmed izvorov urinega takta segrejemo na višjo temperaturo, drugega pa pustimo pri temperaturi 22.5 ℃. Za segrevanje uporabimo toplotni fen. Temperatura segretega izvora se giblje na intervalu [45 ℃, 55 ℃]. Izmerjeni časi so prikazani v tabeli 5.8.

t(min)	0	0.5	1	2	3	4	5	10	15	20
ura (s_0)	-1	-1	-1	-1	-2	-2	-2	-4	-5	-7
ura (s_1)	-1	-1	-1	0	0	0	0	0	0	-1
$TM_{s1} - TM_{s0}$	0	0	0	1	2	2	2	4	5	6

Tabela 5.7: Razlike med urami enot v različnih časovnih točkah.

t(min)	0	0.5	1	2	3	4	5	10
ura (s_0)	-1	-3	-8	-13	-16	-18	-26	-29
ura (s_1)	-1	-1	-1	0	0	0	0	1
$TM_{s1} - TM_{s0}$	0	2	7	13	16	18	26	30

Tabela 5.8: Razlike med urami enot v različnih časovnih točkah s segrevanjem izvora urinega takta ene izmed enot.

Spodnja meja absolutne napake znaša $((TM_{s1} - TM_{s0} + 1) \cdot t_{CLK_0})/2$ ms, kjer je $t_{CLK_0} = 0.0305ms$. Npr. v primeru vrednosti razlike $TM_{s1} - TM_{s0}$, ki je enaka 30, znaša spodnja meja absolutne napake 0,47275 ms. Razliko med obema testoma je bilo za pričakovati, saj je relativna napaka DS32DHZ-oscilatorja na temperaturnem območju [40 ℃, 85 ℃] 7.5 ppm.

Poglavje 6

Zaključek

V okviru diplomskega dela smo uspešno razvili ter implementirali prototip sistema za elektronsko merjenje časa. Izdelali smo matični plošči nadzorne in senzorske enote. Pri tem smo uporabili različne komponente, kot so mikrokontroler serije PIC18 za procesiranje programske kode, temperaturno kompenzirani kvarčni kristal DS32KHZ [7] z visoko natančnostjo kot izvor urinega takta, modul rfm69w [2] za komunikacijo preko radijskih valov z deterministično zakasnitvijo, bluetooth modul rn42 [4] za komuniciranje centralne enote z zunanjimi enotami itn.

Senzorskim enotam smo dodali podporo dveh tipov senzorjev, in sicer vrata z infrardečim žarkom, ki zaznajo objekte oz. subjekte v gibanju in ob tem generirajo prekinitvene dogodke ter tračno stikalo za zaznavanje in generiranje štartnih dogodkov merjenih subjektov. Za sprejemno-oddajni enoti infrardečih žarkov smo izbrali komponenti SFH5110 [6], SFH4510 [7].

Naknadno smo razvili okrnjen operacijski sistem z vsemi za rešitev zahtevanimi elementi. Sem spadajo modul za pošiljanje sporočil naslovljenim objektom, ki omogoča izvajanje njihovih funkcij na različnih prioritetnih nitih, komunikacijski sklad, z nalogo pošiljanja in sprejemanja sporočil med enotami z deterističnimi zakasnitvami, modul z virtualnimi časovniki, modul za upravljanje s časom, grafična knjižnica, gonilniki za uporabo komponent matične plošče itn.

Na zadnje je bila razvita sama aplikacija, in sicer programski del centralne enote vključno z delom na enotah senzorjev.

Skozi celoten razvoj smo imeli glavni povdarek na meritveni natančnosti, pri čemer je bila v orientacijo tudi cena posameznih gradnikov.

Testi so pokazali, da so zastavljeni cilji glede velikosti še dopustne meritvene napake, ki naj bi bila manjša od 1/10 resolucije, doseženi. Resolucija pri merjenju je 1/100 s. Od tu sledi, da je dopustna vrednost napake meritve enaka 1/1000 s. Omeniti velja, da je pričakovanja presegel tudi izmerjen komunikacijski doseg enot.

Torej kot kaže, je projekt za v bodoče na pravi poti.

Ugotavljamo pa, da ima trenutna implementacija tudi pomanjklivosti. Glavni problem, ki se izkaže, je, da je zahtevnost programskega modula na centralni enoti presegla zmogljivosti izbranega mikrokontrolerja.

Velikost notranjega statičnega pomnilnika ne zadostuje zahtevam programske rešitve centralne enote. Zaradi tega je bilo potrebno dodati zunanjo pomnilniško razširitev. Dostopi do razširjenega pomnilnika so za razliko od internega nekajkrat počasnejši. Dodatno je tu še proizvajalčeva cca. 15 odstotna omejitev maksimalne možne frekvence mikrokontrolerja v primeru pomnilniške razširitve. To prinese precejšnje zmanjšanje procesne moči mikrokontrolerja, kar se znatno pozna pri odzivnosti programske rešitve. Posledično je opaziti, da v primeru postavitve prenizkega praga RSSI na rfm69w-modulu programska rešitev preprosto ne deluje več. Razlog temu je, da šum sprejemamo kot podatke, ki jih obdelujemo pri hitrosti 9600 baudov. Za omenjeno obdelavo podatkov porabimo skoraj vso procesno moč tako, da za procesiranje zahtev na glavni niti ne preostane skoraj nič.

Tudi strojno implementiran sklad, ki ga mikrokontroler uporablja za vračanje iz funkcijskih klicev, je pramajhen.

Napake pri časovnem usklajevanju ur bi lahko omejili, če zagotovimo, da je vrednost T_{sync} deljiva s T_{clk_0} in vrednost $SYNC_{offset}$ enaka 0 (slika 5.2).

CRC-koda pri preverjanju paketov je prekratka. Ob sprejemanju šuma se dogaja, da prihaja do sprejemanja napačnih paketov. Z razširitvijo kode

bi zmanjšali verjetnost sprejema takih paketov.

Smernice za v bodoče so naslednje. Počasi se nagibamo k uporabi mikrokontrolerjev bodisi PIC32 bodisi ARM-tehnologije, ki naj bi bili zmogljivejši, z več pomnilnika, več prioritetnimi nivoji izvajanja kode itn.

Napraviti je potrebno elektronska vezja za matične plošče. Sledi izdelava ohišja z anteno ter uporabniškim vmesnikom (morebiti tipkovnica). Nasloviti je potrebno problem vzdrževanja konstantne temperature okoli modula DS32KHZ (podpoglavje 3.2) zavoljo zagotavljanja stabilnega urinega signala.

Porabo električne energije je potrebno omejiti. Ko nastopi čas programske neaktivnosti, je potrebno mikrokontroler in druge komponente postaviti v spanje. Smotrno bi bilo integrirati nek interni baterijski sistem z možnostjo napajanja.

Omogočiti je potrebno enostaven način posodabljanja programske kode na enotah meritvenega sistema.

Kot izvajalec sem projektu posvetil ogromno svojega časa, energije in znanja, zato sem lahko z rezultatom več kot zadovoljen. V bodoče pa bi bilo dobro stopiti korak dlje in poskusiti pripeljati rešitev do končnega produkta.

Literatura

[1] Microchip: PIC18F87K22 Family datasheet

[2] HOPERF: rfm69w ISM transciever module v1.3 datasheet

[3] Maxim integrated: DS32kHz 32.768kHz Temperature-Compensated Crystal Oscillator datasheet

[4] Roving: RN-42/RN-42-N Data Sheet www.rovingnetworks.com DS-RN42-V1.0 12/6/2010

[5] Roving: Bluetooth Data Module Command Reference & Advanced Information User's Guide RN-BT-DATA-UG

[6] Osram: SFH 5110 IR-Receiver for Remote Control Systems datasheet

[7] Osram: GaAs Infrared Emitters (950 nm) in SMR® Package datasheet

[8] H. Partl: *German TEX*, TUGboat Volume 9, Issue 1 (1988)